

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NGOẠI NGỮ – TIN HỌC TP. HỒ CHÍ
MINH KHOA CÔNG NGHỆ THÔNG TIN
★ ☺ ✎



ĐỒ ÁN MÔN HỌC
MẪU THIẾT KẾ CHO PHẦN MỀM
Đề tài: Xây Dựng Trang Web Xem Phim

GIÁO VIÊN HƯỚNG DẪN: Th.S Nguyễn Thị Thúy A.

Nhóm:

Trần Phúc Bình	22DH110370
Nguyễn Thị Thùy Dương	22DH114488
K' Giô Sa Đác	22DH114491
Ngô Dương Kiều Trân	22DH113850

Thành phố Hồ Chí Minh, tháng 3 năm 2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NGOẠI NGỮ – TIN HỌC TP. HỒ CHÍ
MINH KHOA CÔNG NGHỆ THÔNG TIN
★ ☀ ■



ĐỒ ÁN MÔN HỌC
MẪU THIẾT KẾ CHO PHẦN MỀM
Đề tài: Xây Dựng Trang Web Xem Phim

GIÁO VIÊN HƯỚNG DẪN: Th.S Nguyễn Thị Thúy A.

Nhóm:

Trần Phúc Bình	22DH110370
Nguyễn Thị Thùy Dương	22DH114488
K' Giô Sa Đác	22DH114491
Ngô Dương Kiều Trân	22DH113850

Thành phố Hồ Chí Minh, tháng 3 năm 2025

PHIẾU CHẤM ĐIỂM MÔN THI VĂN ĐÁP

 **Điểm phần trình bày – Điểm hệ 10 – Tỷ lệ điểm chiếm 30%**

	CBCT1	CBCT2
Họ tên CBCT Chữ ký: Chữ ký:
Điểm Băng chữ: Băng chữ:
Nhận xét	Ưu điểm :	Nhược điểm :

 **Điểm quá trình – Điểm hệ 10 – Tỷ lệ điểm chiếm 70%**

Họ tên CBCT:

.....

 **Điểm tổng kết: (Băng chữ)**

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến tất cả thầy cô trường đại học Đại Học Ngoại Ngữ - Tin Học TP. Hồ Chí Minh nói chung và các thầy cô trong khoa Công Nghệ Thông Tin nói riêng đã tận tình giảng dạy, truyền đạt những kiến thức và kinh nghiệm quý báu cho chúng em trong suốt quá trình học tập tại trường.

Trong suốt thời gian em làm bài báo cáo đề tài môn Mẫu thiết kế cho phần mềm, em xin gửi lời cảm ơn chân thành và sâu sắc đến thầy ThS. Lương Văn Minh, người thầy đã hết lòng giúp đỡ và theo sát em trong suốt quá trình thực hiện bài đề tài đồ án phần mềm này, chỉ ra cho hướng đi để em có thể hoàn thành tốt nhất bài báo cáo đề tài đồ án phần mềm này đúng thời hạn quy định.

Trong quá trình thực hiện đề tài môn Mẫu thiết kế cho phần mềm, dù em đã cố gắng hoàn thiện đề tài một cách tốt nhất nhưng do thời gian và kiến thức còn hạn chế nên sẽ không tránh khỏi những thiếu sót nhất định, rất mong nhận được sự cảm thông, chia sẻ đến từ những đóng góp ý chân thành từ quý thầy.

Sau cùng em xin gửi lời cảm ơn đến tất cả các bạn và thầy đã tham gia đóng góp ý kiến và giúp đỡ em trong suốt quá trình thực hiện đề tài môn Mẫu thiết kế cho phần mềm.

Chúng em xin chân thành cảm ơn!

Phụ lục viết tắt

PHỤ LỤC VIẾT TẮT

STT	Từ viết tắt	Ý nghĩa
1	CNTT	Công nghệ thông tin
2	DB	Database
3	TMĐT	Thương mại điện tử
4	COD	Cash On Delivery

Bảng 1: Phụ lục viết tắt

MỤC LỤC

GIỚI THIỆU.....	10
MỤC TIÊU ĐỀ TÀI.....	11
1. Giới thiệu.....	11
2. Mục tiêu cụ thể.....	11
3. Kết luận.....	12
1. LÝ THUYẾT:.....	13
1.1. Giới thiệu về C#:.....	13
1.2. Giới thiệu về .NET Framework.....	13
1.3. Giới thiệu về các mẫu thiết kế phần mềm:.....	15
1.3.1. Mẫu Singleton:.....	15
1.3.2. Mẫu Proxy:.....	17
1.3.3. Mẫu Strategy:	18
1.3.4. Mẫu Facade:	18
1.3.5. Mẫu Repository:.....	19
2. XÁC ĐỊNH YÊU CẦU.....	21
2.1. Mô hình cơ cấu tổ chức.....	21
2.1.1. Sơ đồ tổ chức của công ty.....	21
2.1.2. Ý nghĩa các bộ phận	22
2.1.3. Nhu cầu người dùng và Yêu cầu của phần mềm (NGHIỆP VỤ).....	24
2.1.4. Biểu mẫu	27
3. MÔ HÌNH HÓA YÊU CẦU.....	30
3.1. Usecase Diagram.....	30
3.1.1. Sơ đồ mức tổng quát.....	30
3.1.2. Sơ đồ use case Khách (Guest)	31
3.1.3. Sơ đồ use case Người dùng (User).....	32
3.1.4. Sơ đồ use case Quản trị viên (Admin).....	33
3.1.5. Sơ đồ use case Quản lý website (Administrator).....	34
3.1.6. Sơ đồ use case Đăng nhập (Login).....	35
3.1.7. Sơ đồ use case Đăng ký	35
3.1.8. Sơ đồ use case Đặt phim	36

3.2.	Bảng Usecase	37
3.3.	Đặc tả Usecase	38
3.3.1.	Đăng ký tài khoản	38
3.3.2.	Đăng nhập	38
3.3.3.	Sửa thông tin cá nhân	40
3.3.4.	Tìm kiếm phim	42
3.3.5.	Xem phim	43
3.3.6.	Thêm phim vào mục yêu thích	45
3.3.7.	Xóa phim khỏi mục yêu thích	46
3.3.8.	Bình luận, cảm xúc phim	48
3.3.9.	Thêm phim	49
3.3.10.	Xóa phim	50
3.3.11.	Sửa chi tiết phim	52
3.3.12.	Đặt mua phim	54
3.3.13.	Xem chi tiết phim	55
3.3.14.	Thêm tin tức	57
3.3.15.	Thông kê	57
4.	THIẾT KẾ DỮ LIỆU	76
4.1.	Sơ đồ logic	76
4.2.	Chi tiết các bảng	77
4.2.1.	Bảng TaiKhoan (Tài khoản người dùng):.....	77
4.2.2.	Bảng DSPhimBo (Danh sách phim bộ):.....	77
4.2.3.	Bảng CTTapPhim (Chi tiết tập phim bộ).....	78
4.2.4.	Bảng DSPhimLe (Danh sách phim lẻ).....	78
4.2.5.	Bảng HopPhim (Hộp phim xem sau).....	79
4.2.6.	Bảng LichSuPhim (Lịch sử xem phim).....	79
4.2.7.	Bảng TheLoai (Thể loại).....	80
4.2.8.	Bảng QuocGia (Quốc gia).....	80
4.2.9.	Bảng Nam (Năm phát hành phim).....	80
4.2.10.	Bảng Banner (Banner quảng bá phim bộ).....	81
4.2.11.	Bảng tintucphim (Tin tức về phim điện ảnh).....	81
4.2.12.	Bảng gioithieu (Giới thiệu trang website).....	82
4.3.	Các câu lệnh trigger và stored procedures:	83

4.3.1. Trigger:.....	83
4.3.2. Stored Procedures:	84
5. THIẾT KẾ GIAO DIỆN.....	85
5.1.1. Sơ đồ giao diện tổng quát.....	85
5.1.1. Sơ đồ giao diện tổng quát Admin.....	85
5.1.2. Sơ đồ giao diện tổng quát Người xem.....	85
5.1. Giao diện chi tiết.....	86
5.2.1 Người dùng.....	86
5.2.2 Quản trị viên website	88
6. HIỆN THỰC CHƯƠNG TRÌNH.....	122
6.1 Áp dụng mẫu Facade.....	122
6.1.1. Chức năng quản lý tài khoản	122
6.1.2. Chức năng đăng nhập.....	137
6.1.3. Chức năng Tin tức.....	149
6.2 Áp dụng mẫu Strategy	156
6.2.1. Chức năng quản lý thẻ loại và năm phát hành.....	156
6.2.2. Chức năng Tìm kiếm.....	170
6.3 Áp dụng mẫu Repository	176
6.3.1 Chức năng Đăng ký	176
6.3.2 Chức năng Lượt thích.....	190
6.4 Áp dụng mẫu Proxy:.....	202
7. KẾT LUẬN - HƯỚNG PHÁT TRIỂN.....	210
TÀI LIỆU THAM KHẢO.....	211

DANH MỤC HÌNH

Hình 0-1 .NET Framework	14
Hình 2-1 Sơ đồ tổ chức công ty	21
Hình 3-1 UseCase Diagram tổng quát	31
Hình 3-2 Sơ đồ use case Khách (Guest)	32
Hình 3-3 Sơ đồ use case Người dùng (User)	33
Hình 3-4 Sơ đồ use case Quản trị viên (Admin)	34
Hình 3-5 Sơ đồ use case Quản lý website (Administrator)	35
Hình 3-6 Sơ đồ use case Đăng nhập (Login)	36
Hình 3-7 Sơ đồ use case Đăng ký	36
Hình 3-8 Sơ đồ use case Đặt phim	37
Hình 3-9 Quy trình đăng ký tài khoản người dùng	60
Hình 3-10 Quy trình đăng nhập của người dùng	61
Hình 3-11 Quy trình sửa thông tin cá nhân	62
Hình 3-12 Quy trình tìm kiếm phim	63
Hình 3-13 Quy trình xem phim	64
Hình 3-14 Quy trình thêm phim yêu thích	65
Hình 3-15 Quy trình xóa phim yêu thích	65
Hình 3-16 Quy trình bình luận phim	66
Hình 3-17 Quy trình thêm phim	66
Hình 3-18 Quy trình xóa phim	67
Hình 3-19 Quy trình sửa phim	68
Hình 3-20 Quy trình đặt mua phim	69
Hình 3-21 Activity Đăng ký	70
Hình 3-22 Activity Đăng nhập	71
Hình 3-23 Activity tìm kiếm và lọc phim	72
Hình 3-24 Activity cho chọn phim	73
Hình 3-25 Activity Thanh toán	74
Hình 3-26 Activity đánh giá và bình luận	75
Hình 3-27 Activity Quản lý danh sách phim yêu thích	75
Hình 3-28 Class Diagram	77
Hình 4-1 Sơ đồ logic	78
Hình 4-2 Trigger	84
Hình 4-3 Stored Procedures:	85
Hình 5-1 Sơ đồ giao diện tổng quát Admin	86
Hình 5-2 Sơ đồ giao diện tổng quát Người xem	86
Hình 5-3 Giao diện trang chủ website xem phim của người dùng	87
Hình 5-4 Giao diện đăng nhập tài khoản vào hệ thống	88
Hình 5-5 Giao diện đăng ký tài khoản vào hệ thống	89
Hình 5-6 Giao diện trang chủ của người quản trị (Admin và Administrator)	90
Hình 5-7 Giao diện menu quản lý danh mục của người quản trị	91
Hình 5-8 Giao diện trang chủ	91
Hình 5-9 Giao diện kiểm tra nhập liệu	92
Hình 5-10 Giao diện trang chủ	93
Hình 5-11 Giao diện đăng nhập sai	94
Hình 5-12 Giao diện đăng ký	95
Hình 5-13 Giao diện đăng ký tài khoản trùng	96

Hình 5-14 Giao diện tìm kiếm phim bằng tên phim (Người dùng)	98
Hình 5-15 Giao diện tìm kiếm phim bằng tên phim (Người dùng)	98
Hình 5-16 Giao diện tìm kiếm phim theo thể loại phim	99
Hình 5-17 Giao diện tìm kiếm phim theo năm phát hành.....	99
Hình 5-18 Giao diện tìm kiếm phim theo quốc gia.....	100
Hình 5-19 Giao diện kết quả tìm kiếm.....	101
Hình 5-20 Giao diện kết quả tìm kiếm.....	101
Hình 5-21 Giao diện kết quả tìm kiếm.....	102
Hình 5-22 Giao diện hiển thị danh sách các phim bộ và phim lẻ ở trang chủ	102
Hình 5-23 Giao diện hiển thị danh sách phim xem sau của người dùng	104
Hình 5-24 Giao diện hiển thị các phim đã xem của người dùng (lịch sử xem phim).....	104
Hình 5-25 Giao diện đăng xuất.....	105
Hình 5-26 Giao diện danh sách phim đã được thêm vào yêu thích	106
Hình 5-27 Giao diện danh sách phim đã được đặt mua đang chờ thanh toán.....	107
Hình 5-28 Giao diện hiển thị thống kê top phim theo lượt xem, phim mới.....	108
Hình 5-29 Giao diện quản lý danh sách tài khoản (mã hóa mật khẩu).....	109
Hình 5-30 Giao diện quản lý danh sách các phim bộ	111
Hình 5-31 Giao diện quản lý danh sách các phim lẻ	114
Hình 5-32 Giao diện quản lý danh sách các tin tức của phim.....	115
Hình 5-33 Giao diện quản lý danh sách các hình ảnh banner quảng cáo của phim.....	117
Hình 5-34 Giao diện quản lý danh sách các quốc gia sản xuất phim	118
Hình 5-35 Giao diện quản lý danh sách thể loại phim	119
Hình 5-36 Giao diện quản lý danh sách năm phát hành các phim	120
Hình 5-37 Giao diện quản lý thống kê tổng quát về phim	122
Hình 6-1 Sơ đồ quản lý tài khoản	128
Hình 6-2 Sơ đồ đăng nhập	145
Hình 6-3 Sơ đồ tin tức	156
Hình 6-4 Sơ đồ quản lý thể loại	158
Hình 6-5 Sơ đồ lượt yêu thích	202
Hình 6-6 Sơ đồ lượt thích	210

DANH MỤC BẢNG

Bảng 2-1 Ý nghĩa các bộ phận.....	22
Bảng 2-2 Nhu cầu người dùng và yêu cầu phần mềm.....	23
Bảng 2-3 BM01: Đăng ký tài khoản.....	27
Bảng 2-4 BM02: Đánh giá.....	27
Bảng 2-5 BM03: Lịch sử xem phim.....	28
Bảng 2-6 BM04: Lấy lại mật khẩu.....	29
Bảng 2-7 BM05: Đăng ký nâng cấp tài khoản lên gói trả phí.....	30
Bảng 3-1 Bảng Usecase	38
Bảng 3-2 UC Đăng ký tài khoản.....	39
Bảng 3-3 UC Đăng nhập.....	40
Bảng 3-4 UC Sửa thông tin cá nhân.....	40
Bảng 3-5 UC Tìm kiếm phim.....	43
Bảng 3-6 UC Xem phim.....	44
Bảng 3-7 UC Thêm phim vào mục yêu thích.....	46
Bảng 3-8 UC Xóa phim khỏi mục yêu thích.....	47
Bảng 3-9 UC Bình luận, cảm xúc phim.....	49
Bảng 3-10 UC Thêm phim.....	50
Bảng 3-11 UC Xóa phim	51
Bảng 3-12 UC Sửa chi tiết phim.....	54
Bảng 3-13 UC Đặt mua phim.....	56
Bảng 3-14 UC Xem chi tiết phim.....	57
Bảng 3-15 UC Thêm tin tức.....	58
Bảng 3-16 UC Thống kê.....	58
Bảng 4-1 Bảng TaiKhoan (Tài khoản người dùng):.....	79
Bảng 4-2 Bảng DSPhimBo (Danh sách phim bộ):.....	79
Bảng 4-3 Bảng CTTapPhim (Chi tiết tập phim bộ).....	80
Bảng 4-4 Bảng DSPhimLe (Danh sách phim lẻ).....	80
Bảng 4-5 Bảng HopPhim (Hộp phim xem sau).....	81
Bảng 4-6 Bảng LichSuPhim (Lịch sử xem phim).....	81
Bảng 4-7 Bảng TheLoai (Thể loại).....	81
Bảng 4-8 Bảng QuocGia (Quốc gia).....	82
Bảng 4-9 Bảng Nam (Năm phát hành phim).....	82
Bảng 4-10 Bảng Banner (Banner quảng bá phim bộ).....	82
Bảng 4-11 Bảng tintucphim (Tin tức về phim điện ảnh).....	82
Bảng 4-12 Bảng gioithieu (Giới thiệu trang website).....	83

KẾ HOẠCH – PHÂN CÔNG NHÓM

MSSV	HỌ VÀ TÊN	MỨC ĐỘ HOÀN THÀNH
22DH113850	Ngô Dương Kiều Trần	100%
22DH110370	Trần Phúc Bình	100%
22DH114488	Nguyễn Thị Thùy Dương	100%
22DH114491	K' Giô Sa Đác	100%

GIỚI THIỆU

Ngày nay, hình thức bán hàng trực tuyến qua mạng với nhiều loại mặt hàng đa dạng rất phổ biến và ra đời. Cũng giống như các mặt hàng khác trên thị trường hiện nay, thời trang cũng là một vấn đề được nhiều người quan tâm kể cả nam và nữ, vì vậy giày dép đang là thứ mà khách hàng có nhu cầu mua hàng rất cao, tuy nhiên đa số họ vẫn yêu thích xu hướng mua hàng trực tuyến và tham khảo các hình ảnh sản phẩm, giá cả thông qua website trước khi đi đến cửa hàng, hoặc mua và thanh toán trực tuyến trên website. Nên khách hàng muốn xây dựng website bán hàng thu hút, ấn tượng sẽ giúp quảng bá được cửa hàng góp phần phát triển và thành công.

Xây dựng một website bán hàng online là một trong những công việc phổ biến và quan trọng của các doanh nghiệp trong kỷ nguyên số hiện nay. Với việc ngày càng nhiều người mua sắm trực tuyến, việc có một trang web bán hàng chuyên nghiệp và hiệu quả là điều vô cùng quan trọng để tăng doanh số bán hàng và nâng cao uy tín thương hiệu.

Với xu hướng công nghệ thông tin thời đại 4.0, khi mà mọi gia đình, cá nhân đều dễ dàng truy cập internet từ thiết bị smartphone, laptop, smart TV. Nên việc bán hàng trên internet là một việc cần để bắt kịp xu thế và chiếm thị phần. Chủ cửa hàng cần một trang web để họ có thể quảng bá sản phẩm trên đó.

Việc xây dựng một trang web bán hàng online tốt cần phải có thiết kế hấp dẫn, giao diện thân thiện với người dùng, chức năng đầy đủ và dễ sử dụng. Điều này sẽ giúp cho khách hàng tiết kiệm thời gian, dễ dàng tìm kiếm sản phẩm mình cần một cách nhanh chóng và tiện lợi, khách hàng sẽ có nhiều sự lựa chọn và dễ dàng so sánh giá cả sản phẩm giữa các website bán hàng trực tuyến với nhau. Bên cạnh đó, trang web cần có hệ thống quản lý đơn hàng và thanh toán đáng tin cậy, bảo mật thông tin khách hàng và hỗ trợ khách hàng nhanh chóng.

Tóm lại, xây dựng một trang web bán hàng online là một công việc quan trọng và cần thiết để giúp các doanh nghiệp tăng cường doanh số bán hàng và nâng cao uy tín thương hiệu. Với sự phát triển của kỷ nguyên số, việc có một trang web bán hàng online chuyên nghiệp và hiệu quả là điều cần thiết để cạnh tranh và phát triển trong thị trường kinh doanh ngày càng cạnh tranh.

MỤC TIÊU ĐỀ TÀI

1. Giới thiệu

Trong thời đại công nghệ số, nhu cầu giải trí trực tuyến, đặc biệt là xem phim, ngày càng trở nên phổ biến. Việc xây dựng một website xem phim giúp người dùng dễ dàng tiếp cận kho phim đa dạng, chất lượng cao mà không cần tải xuống. Đề tài này hướng đến việc thiết kế và phát triển một nền tảng xem phim trực tuyến đáp ứng nhu cầu người dùng về giao diện thân thiện, tốc độ nhanh và trải nghiệm xem mượt mà.

2. Mục tiêu cụ thể

- Cung cấp nền tảng xem phim chuyên nghiệp
 - Xây dựng hệ thống quản lý phim, thể loại phim, diễn viên và đánh giá.
 - Hỗ trợ phát video trực tuyến với chất lượng cao.
- Giao diện thân thiện, tối ưu trải nghiệm người dùng
 - Thiết kế UI/UX đơn giản, dễ sử dụng trên cả máy tính và điện thoại.
 - Hỗ trợ tìm kiếm phim theo nhiều tiêu chí (tên phim, thể loại, năm phát hành, diễn viên).
- Quản lý tài khoản người dùng
 - Hỗ trợ đăng ký, đăng nhập, quản lý danh sách phim yêu thích.
 - Cho phép người dùng bình luận, đánh giá phim.
- Phát triển hệ thống gợi ý phim thông minh (Nâng cao)
 - Đề xuất phim dựa trên sở thích và lịch sử xem của người dùng.
- Tích hợp hệ thống thanh toán cho tài khoản VIP (Thêm)
 - Cung cấp dịch vụ xem phim không quảng cáo và chất lượng cao hơn cho người dùng trả phí.
- Tăng cường bảo mật và hiệu suất hệ thống
 - Đảm bảo dữ liệu người dùng và nội dung phim được bảo vệ an toàn.
 - Tối ưu hóa tốc độ tải trang và phát video mượt mà.

3. Kết luận

Với mục tiêu trên, website xem phim không chỉ mang lại trải nghiệm giải trí tốt nhất cho người dùng mà còn giúp nhà phát triển có một hệ thống ổn định, dễ mở rộng trong tương lai.

1. LÝ THUYẾT:

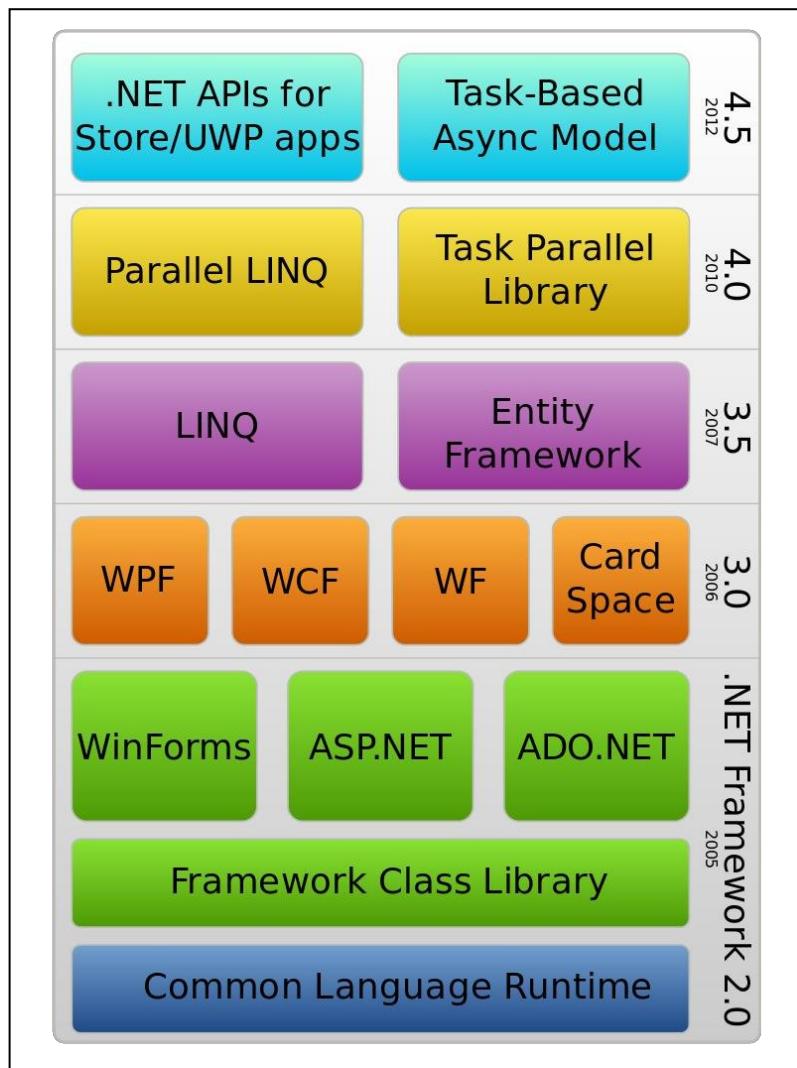
1.1. Giới thiệu về C#:

- C# là một ngôn ngữ lập trình hiện đại, được thiết kế bởi Microsoft và ra mắt vào năm 2000. C# là một ngôn ngữ lập trình đa mục đích, được sử dụng để phát triển các ứng dụng máy tính, ứng dụng web và các ứng dụng di động trên nền tảng Microsoft. C# được thiết kế để có tính năng an toàn và dễ dàng sử dụng, cung cấp một số tính năng mạnh mẽ cho người lập trình như:
 - Hỗ trợ các kiểu dữ liệu đối tượng và lớp.
 - Hỗ trợ quản lý bộ nhớ tự động.
 - Hỗ trợ tính kế thừa và đa hình.
 - Hỗ trợ phát triển ứng dụng đa luồng.
 - Hỗ trợ thao tác với các tệp tin, định dạng XML và cơ sở dữ liệu.

- C# là một trong những ngôn ngữ lập trình phổ biến nhất trên thế giới và được sử dụng rộng rãi trong các ứng dụng như game, ứng dụng văn phòng, ứng dụng web, ứng dụng di động và nhiều lĩnh vực khác. C# được hỗ trợ bởi Visual Studio, một môi trường phát triển tích hợp được cung cấp bởi Microsoft để giúp người lập trình phát triển các ứng dụng C# dễ dàng hơn.

1.2. Giới thiệu về .NET Framework

- .NET Framework là một nền tảng lập trình và thực thi ứng dụng chủ yếu trên hệ điều hành Microsoft Windows. Nó cung cấp một môi trường phần mềm gọi là Common Language Runtime (CLR) để triển khai các chương trình phần mềm và hỗ trợ việc xây dựng các chương trình phần mềm với nhiều thư viện lập trình sẵn như lập trình giao diện, truy cập cơ sở dữ liệu, ứng dụng web, các giải thuật, cấu trúc dữ liệu, giao tiếp mạng, v.v. Nó đơn giản hóa việc viết ứng dụng bằng cách cung cấp các thành phần được thiết kế sẵn và hỗ trợ bởi nhiều công cụ và IDE. Microsoft đã phát triển Visual Studio để hỗ trợ xây dựng ứng dụng .NET.



Hình 0-1 .NET Framework

- Lịch sử phát triển:
 - .NET Framework là một nền tảng lập trình và thực thi ứng dụng được phát triển bởi Microsoft từ năm 2000. Ban đầu, .NET Framework được phát triển nhằm mục đích thay thế các công nghệ lập trình cũ như Microsoft Visual Basic và Microsoft Visual C++. Nền tảng này được xây dựng trên nền tảng COM (Component Object Model) và được phát hành với tên gọi .NET Framework 1.0 vào tháng 2 năm 2002.
 - Sau đó, .NET Framework 1.1 được phát hành vào tháng 4 năm 2003, với các tính năng bổ sung và cải tiến về an ninh và hiệu suất.
 - Phiên bản tiếp theo là .NET Framework 2.0, được phát hành vào năm 2005, với sự bổ sung của Windows Presentation Foundation (WPF) và Windows Communication Foundation (WCF).

- .NET Framework 3.0 được phát hành vào năm 2006 với sự bổ sung của Windows Workflow Foundation (WF) và Windows CardSpace.
- Sau đó, .NET Framework 3.5 được phát hành vào năm 2007 với nhiều cải tiến về các tính năng của .NET Framework 2.0.
- .NET Framework 4 được phát hành vào năm 2010 với sự bổ sung của Dynamic Language Runtime (DLR), Parallel Extensions và Managed Extensibility Framework (MEF).
- Sau đó, .NET Framework 4.5 được phát hành vào năm 2012, với sự bổ sung của các tính năng mới như RyuJIT, Async/Await, và nhiều cải tiến về hiệu suất và an ninh.
- .NET Framework 4.6, 4.7 và 4.8 được phát hành trong các năm 2015, 2017 và 2019 lần lượt, với sự cải tiến về hiệu suất, an ninh và tính năng.
- Tuy nhiên, Microsoft đã chuyển sang phát triển .NET Core là một phiên bản độc lập, có khả năng chạy trên nhiều nền tảng và được phát hành với tên .NET 5 vào năm 2020, và được cập nhật tiếp theo với .NET 6 vào năm 2021. Các phiên bản .NET Core không hoàn toàn tương thích với .NET Framework, tuy nhiên, các công cụ và thư viện được cung cấp để chuyển đổi các ứng dụng từ .NET Framework sang .NET Core.

1.3. Giới thiệu về các mẫu thiết kế phần mềm:

1.3.1. Mẫu Singleton:

Singleton là một mẫu thiết kế thuộc nhóm mẫu thiết kế hành vi (behavioral design patterns) trong lập trình hướng đối tượng. Mục đích chính của mẫu thiết kế Singleton là đảm bảo rằng một lớp chỉ có duy nhất một thể hiện (instance) và cung cấp một điểm truy cập toàn cục đến thể hiện đó.

Dưới đây là một ví dụ về cách triển khai Singleton trong ngôn ngữ lập trình C#:

```
public sealed class Singleton
{
    private static Singleton instance;
    private static readonly object lockObject = new object();
    private Singleton() { }

    public static Singleton Instance
    {
        get
        {
            if (instance == null)
            {
                lock (lockObject)
                {
                    if (instance == null)
                    {
                        instance = new Singleton();
                    }
                }
            }
            return instance;
        }
    }

    public void DoSomething()
    {
        Console.WriteLine("Singleton is doing something...");
    }
}
```

1.3.2. Mẫu Proxy:

Mẫu thiết kế Proxy (Đại diện) là một mẫu cấu trúc (Structural Pattern) được sử dụng để cung cấp một đối tượng thay thế hoặc trung gian cho một đối tượng thực tế. Proxy kiểm soát quyền truy cập vào đối tượng gốc, giúp tăng cường tính bảo mật, tối ưu hiệu suất hoặc cung cấp chức năng bổ sung.

```
//proxy
private void Load ViewData()
{
    ViewData["TheLoai"] = _data.GetTheLoais();
    ViewData["Nam"] = _data.GetNams();
    ViewData["QuocGia"] = _data.GetQuocGias();
}

public ActionResult XemPhim(int id, int? tap)
{
    Load ViewData();
    int pageSize = 1;
    int pageNum = tap ?? 1;
    var tapPhims = _data.GetCTTapPhims(id, null, 1000); // Lấy tất cả tập
    var currentTap = _data.GetCTTapPhims(id, tap, 1); // Lấy tập hiện tại
    ViewData["p"] = currentTap;
    return View(tapPhims.ToPagedList(pageNum, pageSize));
}

public ActionResult XemPhimLe(int id)
{
    Load ViewData();
    var phim = _data.GetDSPhimLes(id);
    return View(phim);
}
```

1.3.3. Mẫu Strategy:

Mẫu thiết kế Strategy là một mẫu thiết kế hành vi cho phép bạn định nghĩa một nhóm các thuật toán, đặt mỗi thuật toán vào một lớp riêng biệt và làm cho các đối tượng của chúng có thể hoán đổi được.

```
namespace TMDT.Patterns
{
    public class Strategy
    {
        public interface IVoucherValidationStrategy
        {
            bool Validate(VOUCERSOP voucher);
        }
        public class StartDateValidationStrategy : IVoucherValidationStrategy
        {
            public bool Validate(VOUCERSOP voucher)
            {
                // Implement logic to validate the start date of the voucher
                DateTime currentDate = DateTime.now.Date;
                return voucher.GAYBD > currentDate;
            }
        }
        public class EndDateValidationStrategy : IVoucherValidationStrategy
        {
            public bool Validate(VOUCERSOP voucher)
            {
                // Implement logic to validate the end date of the voucher
                currentDate = DateTime.now.Date;
                return voucher.GAYKT < currentDate;
            }
        }
    }
}
```

1.3.4. Mẫu Facade:

Facade Pattern (mẫu Facade) là một mẫu thiết kế cấu trúc (structural design pattern) giúp ẩn đi sự phức tạp của hệ thống bên trong, cung cấp một giao diện đơn giản và dễ sử dụng để giao tiếp với các hệ thống phức tạp. Nó hoạt động như một "mặt tiền" cho các lớp khác, nhằm tách biệt người dùng khỏi các chi tiết triển khai phức tạp. Facade Pattern giúp giảm sự phức tạp của hệ thống bằng cách cung cấp một lớp đơn giản với các phương thức dễ hiểu để giao tiếp với các lớp phức tạp hoặc nhiều lớp trong hệ thống.

```

namespace TMĐT.Patterns
{
    // Interface chiến lược cho xác thực voucher
    public interface IVoucherValidationStrategy
    {
        bool Validate(Voucher voucher);
    }

    // Lớp chiến lược để xác thực ngày bắt đầu của voucher
    public class StartDateValidationStrategy : IVoucherValidationStrategy
    {
        public bool Validate(Voucher voucher)
        {
            // Implement logic to validate the start date of the voucher
            DateTime currentDate = DateTime.Now;
            return voucher.StartDate <= currentDate;
        }
    }
}

```

```

// Lớp chiến lược để xác thực ngày kết thúc của voucher
public class EndDateValidationStrategy : IVoucherValidationStrategy
{
    public bool Validate(Voucher voucher)
    {
        // Implement logic to validate the end date of the voucher
        DateTime currentDate = DateTime.Now;
        return voucher.EndDate >= currentDate;
    }
}

// Lớp chứa thông tin voucher
public class Voucher
{
    public DateTime StartDate { get; set; }
    public DateTime EndDate { get; set; }

    public Voucher(DateTime startDate, DateTime endDate)
    {
        StartDate = startDate;
        EndDate = endDate;
    }
}

```



1.3.5. Mẫu Repository:

Repository Pattern là một mẫu thiết kế (design pattern) thuộc nhóm Data Access Patterns (Mẫu tiếp cận dữ liệu). Mục tiêu chính của mẫu này là cung cấp một lớp trung gian giữa ứng dụng và cơ sở dữ liệu hoặc bất kỳ nguồn lưu trữ dữ liệu nào. Nó giúp ẩn đi chi tiết việc truy cập cơ sở dữ liệu và tạo một giao diện thống nhất cho việc thao tác với dữ liệu.

```
namespace TMDT.Patterns
{
    // Lớp thực thi Repository để thao tác với cơ sở dữ liệu
    public class UserRepository : IUserRepository
    {
        // Giả sử có một cơ sở dữ liệu giả lập (có thể là một danh sách người dùng)
        private List<User> _users = new List<User>();

        // Lấy người dùng theo Id
        public User GetById(int id)
        {
            return _users.FirstOrDefault(u => u.Id == id);
        }

        // Thêm người dùng vào cơ sở dữ liệu
        public void Add(User user)
        {
            _users.Add(user);
        }

        // Cập nhật thông tin người dùng
        public void Update(User user)
        {
            var existingUser = _users.FirstOrDefault(u => u.Id == user.Id);
            if (existingUser != null)
            {
                existingUser.Username = user.Username;
                existingUser.Password = user.Password;
            }
        }

        // Xóa người dùng khỏi cơ sở dữ liệu
        public void Delete(int id)
        {
            var user = _users.FirstOrDefault(u => u.Id == id);
            if (user != null)
            {
                _users.Remove(user);
            }
        }
    }
}
```



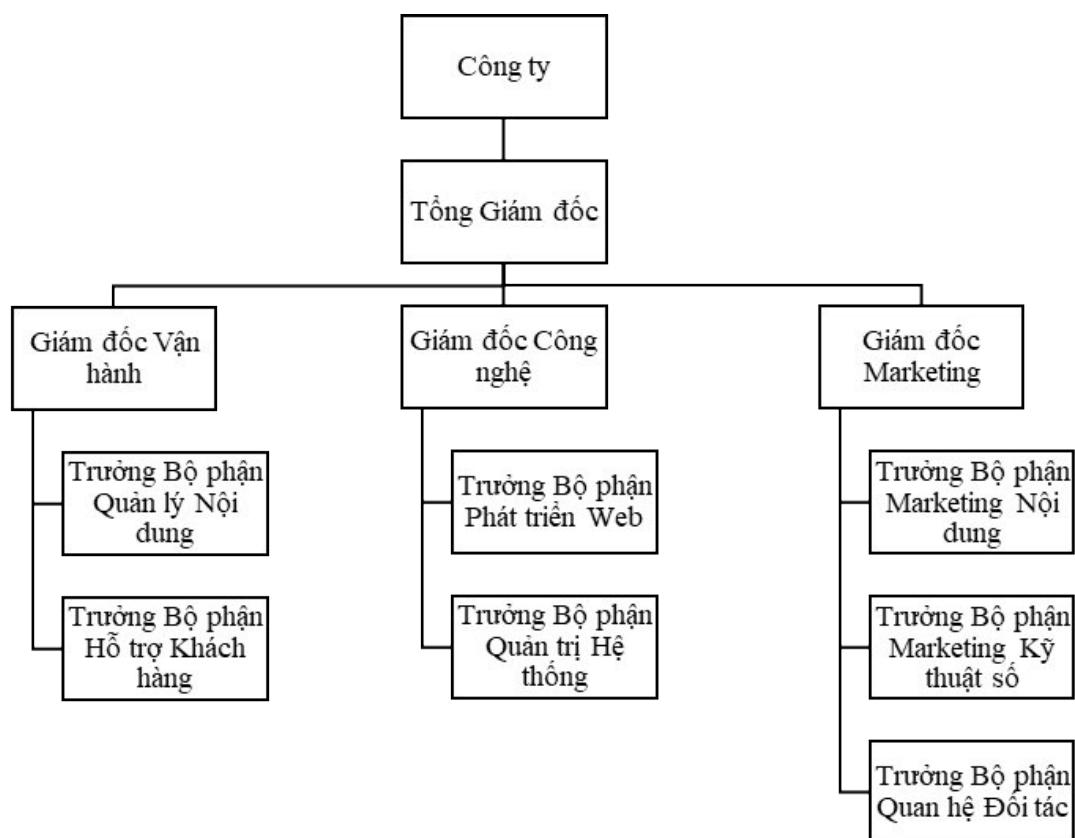
2. XÁC ĐỊNH YÊU CẦU

Một website xem phim là một nền tảng trực tuyến cho phép người dùng truy cập và thưởng thức các bộ phim mà không cần đến rạp chiếu phim hay cửa hàng băng đĩa.

Trên đó, nhà cung cấp nội dung có thể đăng tải thông tin về các bộ phim, bao gồm trailer, mô tả nội dung, diễn viên, đạo diễn, đánh giá và các thông tin liên quan mà không cần hệ thống phân phối vật lý. Đồng thời, người xem có thể tiếp cận và lựa chọn phim một cách trực quan, dễ dàng, chỉ cần có kết nối Internet.

2.1. Mô hình cơ cấu tổ chức

2.1.1. Sơ đồ tổ chức của công ty



Hình 2-1 Sơ đồ tổ chức công ty

2.1.2. Ý nghĩa các bộ phận

Bảng 2-1 Ý nghĩa các bộ phận

STT	Vai trò (role)	Mô tả
1	Tổng Giám đốc	<ul style="list-style-type: none"> Người đứng đầu công ty, chịu trách nhiệm về tầm nhìn, chiến lược và hiệu quả hoạt động tổng thể. Đưa ra quyết định quan trọng, quản lý nguồn lực và đại diện cho công ty trước các bên liên quan.
2	Giám đốc Vận hành	<ul style="list-style-type: none"> Quản lý các hoạt động hàng ngày của công ty, đảm bảo vận hành trơn tru và hiệu quả. Giám sát các bộ phận Quản lý Nội dung và Hỗ trợ Khách hàng.
3	Trưởng Bộ phận Quản lý Nội dung	<ul style="list-style-type: none"> Quản lý và cập nhật nội dung trên website (phim, tin tức, v.v.). Đảm bảo chất lượng và tính phù hợp của nội dung. Tương ứng với vai trò Actor Quản trị viên (Admin).
4	Trưởng Bộ phận Hỗ trợ Khách hàng	<ul style="list-style-type: none"> Giải đáp thắc mắc và hỗ trợ người dùng giải quyết vấn đề. Quản lý các kênh hỗ trợ khách hàng (email, điện thoại, chat). Đảm bảo sự hài lòng của người dùng.
5	Giám đốc Công nghệ	<ul style="list-style-type: none"> Quản lý và phát triển hệ thống kỹ thuật của website. Giám sát các bộ phận Phát triển Web và Quản trị Hệ thống.
6	Trưởng Bộ phận Phát triển Web	<ul style="list-style-type: none"> Xây dựng, phát triển và bảo trì website.

		<ul style="list-style-type: none"> Nghiên cứu và áp dụng công nghệ mới để cải thiện trải nghiệm người dùng.
7	Trưởng Bộ phận Quản trị Hệ thống	<ul style="list-style-type: none"> Đảm bảo hệ thống hoạt động ổn định, an toàn và bảo mật. Quản lý server, cơ sở dữ liệu và các hạ tầng kỹ thuật khác.

8	Giám đốc Marketing	<ul style="list-style-type: none"> Xây dựng và triển khai chiến lược marketing để thu hút và giữ chân người dùng. Giám sát các bộ phận Marketing Nội dung, Marketing Kỹ thuật số và Quan hệ Đối tác.
9	Trưởng Bộ phận Marketing Nội dung	<ul style="list-style-type: none"> Tạo và quản lý nội dung marketing hấp dẫn và chất lượng. Xây dựng chiến lược nội dung để tăng tương tác và nhận diện thương hiệu.
10	Trưởng Bộ phận Marketing Kỹ thuật số	<ul style="list-style-type: none"> Thực hiện các chiến dịch marketing trực tuyến (SEO, quảng cáo trực tuyến, mạng xã hội, v.v.). Phân tích dữ liệu và tối ưu hóa hiệu quả các chiến dịch.
11	Trưởng Bộ phận Quan hệ Đối tác	<ul style="list-style-type: none"> Xây dựng và duy trì quan hệ với các đối tác (nhà cung cấp phim, nhà quảng cáo, v.v.). Tìm kiếm và phát triển các cơ hội hợp tác mới.

2.1.3. Nhu cầu người dùng và Yêu cầu của phần mềm (NGHIỆP VỤ)

Bảng 2-2 Nhu cầu người dùng và yêu cầu phần mềm

ST T	Nhu cầu	Nghiệp vụ	Ai thực hiện
			Bộ phận thực hiện
1	Tìm kiếm phim dễ dàng theo keyword.	Tìm kiếm theo từ khóa.	Người dùng.
2	Đánh giá, bình luận phim	Viết đánh giá, bình luận.	Người dùng.
3	Chia sẻ phim với bạn bè qua mạng xã hội	Chia sẻ link phim, thông tin phim	Người dùng.
4	Xem phim trên nhiều thiết bị (điện thoại, máy tính bảng, TV)	Đồng bộ tài khoản, tối ưu hóa giao diện cho từng thiết bị	Bộ phận phát triển hệ thống
5	Người dùng muốn đăng ký tài khoản	Đăng ký tài khoản	Người dùng. .
6	Người dùng muốn đăng nhập vào website bằng tài khoản	Đăng nhập vào website	Người dùng.
7	Người dùng muốn xem đánh giá của phim.	Xem đánh giá người khác từng đánh giá	Người dùng.
8	Lọc phim theo thể loại (hành động, hài, kinh dị, v.v.)	Lọc theo dropdown	Người dùng.
9	Lọc phim theo năm sản xuất	Lọc theo dropdown	Người dùng.
10	Lọc phim theo quốc gia sản xuất	Lọc theo dropdown	Người dùng.

11	Hiển thị danh sách phim mới phát hành	Hiển thị danh sách phim mới nhất trên trang chủ	Bộ phận phát triển hệ thống.
12	Hiển thị danh sách phim được xem nhiều nhất	Hiển thị danh sách phim xem nhiều nhất trên trang chủ	Bộ phận phát triển hệ thống.
13	Thêm phim vào danh sách yêu thích	Thêm phim vào danh sách yêu thích	Người dùng.
14	Phát phim trực tuyến với chất lượng hình ảnh và âm thanh cao	Phát phim với chất lượng hình ảnh và âm thanh cao	Bộ phận phát triển hệ thống.
15	Hỗ trợ nhiều độ phân giải (SD, HD, Full HD, 4K)	Cung cấp nhiều độ phân giải khi phát phim	Bộ phận phát triển hệ thống.
16	Hỗ trợ nhiều ngôn ngữ phụ đề	Cung cấp nhiều lựa chọn ngôn ngữ phụ đề	Bộ phận phát triển hệ thống.
17	Điều chỉnh âm lượng, độ sáng, tua nhanh, tua chậm	Cung cấp các công cụ điều chỉnh âm lượng, độ sáng, tua phim	Bộ phận phát triển hệ thống.
18	Xem phim ở chế độ toàn màn hình	Cho phép người dùng xem phim ở chế độ toàn màn hình	Bộ phận phát triển hệ thống.
19	Tiếp tục xem phim từ lần trước	Lưu trạng thái phim và cho phép tiếp tục xem từ lần trước	Bộ phận phát triển hệ thống.
20	Tải phim về xem offline	Cho phép tải phim về xem offline	Bộ phận phát triển hệ thống.
21	Quản lý lịch sử xem	Cung cấp chức năng quản lý lịch sử xem phim	Bộ phận phát triển hệ thống.

22	Quản lý danh sách yêu thích	Quản lý danh sách yêu thích phim của người dùng	Bộ phận phát triển hệ thống.
26	Nâng cấp tài khoản lên gói trả phí	Cung cấp tùy chọn nâng cấp tài khoản lên gói trả phí	Bộ phận hỗ trợ khách hàng
27	Hủy đăng ký tài khoản	Xử lý yêu cầu hủy tài khoản	Bộ phận hỗ trợ khách hàng.
28	Người dùng muốn đăng nhập vào website Google	Đăng nhập vào website Bằng Google	Người dùng.
29	Người dùng muốn đăng nhập vào website bằng FaceBook	Đăng nhập vào website bằng Facebook	Người dùng.

2.1.4. Biểu mẫu

a. BM01: Đăng ký tài khoản

Bảng 2-3 BM01: Đăng ký tài khoản

ĐĂNG KÝ TÀI KHOẢN	
BM01	
Họ và tên:.....	Mật khẩu:.....
Số điện thoại:.....	Xác nhận mật khẩu:
Email:.....	

QĐ1: + Thông tin cá nhân: Họ và tên, số điện thoại, email là bắt buộc.

- + Mật khẩu: phải có ít nhất 8 ký tự, bao gồm cả chữ hoa, chữ thường và số.
- + Xác nhận mật khẩu: phải trùng khớp với mật khẩu đã nhập.

b. BM02: Đánh giá

Bảng 2-4 BM02: Đánh giá

KHÁCH HÀNG MUỐN ĐÁNH GIÁ	
BM02	
Họ và tên:.....	Nhận xét về trải nghiệm xem phim:...
Số điện thoại:.....	Đánh giá số sao:.....
Email:.....	Lưu thông tin đánh giá:.....

QĐ2: + Thông tin bắt buộc: Tên khách hàng và đánh giá là bắt buộc.

- + Đánh giá số sao: Bắt buộc phải chọn số sao để đánh giá.

c. BM03: Lịch sử xem phim

Bảng 2-5 BM03: Lịch sử xem phim

BM12:		Lịch sử phim đã xem
STT	Tên phim	Thời gian
1		
2		

QĐ3

- + Tất cả lịch sử của khách hàng sẽ được ghi nhận lại
- + Lịch sử sẽ được lưu trữ trong 12 tháng

d. BM04: Lấy lại mật khẩu

Bảng 2-6 BM04: Lấy lại mật khẩu

BM10	LẤY LẠI MẬT KHẨU
Email:.....	
Số điện thoại:.....	
Nhập mật khẩu mới:.....	
Nhập lại mật khẩu mới:.....	

Lưu Ý:

- Email: Địa chỉ email đã đăng ký sẽ được sử dụng để gửi liên kết đặt lại mật khẩu.
- Số điện thoại: Tùy chọn, có thể cung cấp để hỗ trợ việc xác minh và đảm bảo tính bảo mật.

QĐ4:

- + Thông Tin Bắt Buộc: Email là thông tin bắt buộc để sử dụng tính năng "Quên mật khẩu".
- + Yêu Cầu Thêm: Số điện thoại là tùy chọn nhưng có thể cung cấp thêm thông tin cho quá trình xác minh.

+Yêu cầu Mật khẩu mới:

-Ít nhất 8 ký tự,bao gồm chữ hoa,chữ thường,số và ký tự đặc biệt.

-Không trùng với mật khẩu cũ.

+Mật Thông Tin: Thông tin cá nhân được bảo mật và chỉ sử dụng cho mục đích đặt lại mật khẩu.

+Bán Tự Động: Hệ thống sẽ tự động xử lý yêu cầu và gửi liên kết đặt lại mật khẩu đến email đã cung cấp.

e. BM05: Đăng ký nâng cấp tài khoản lên gói trả phí

Bảng 2-7 BM05: Đăng ký nâng cấp tài khoản lên gói trả phí

BM15	Biểu mẫu đăng ký tài khoản đối tác
Tên đăng nhập:.....	
Mật khẩu:.....	
Xác nhận mật khẩu:.....	
Thêm mô tả	
Họ và tên:.....	
Số điện thoại:.....	
Email:.....	
Địa chỉ:.....	
Loại nhà phí:.....	

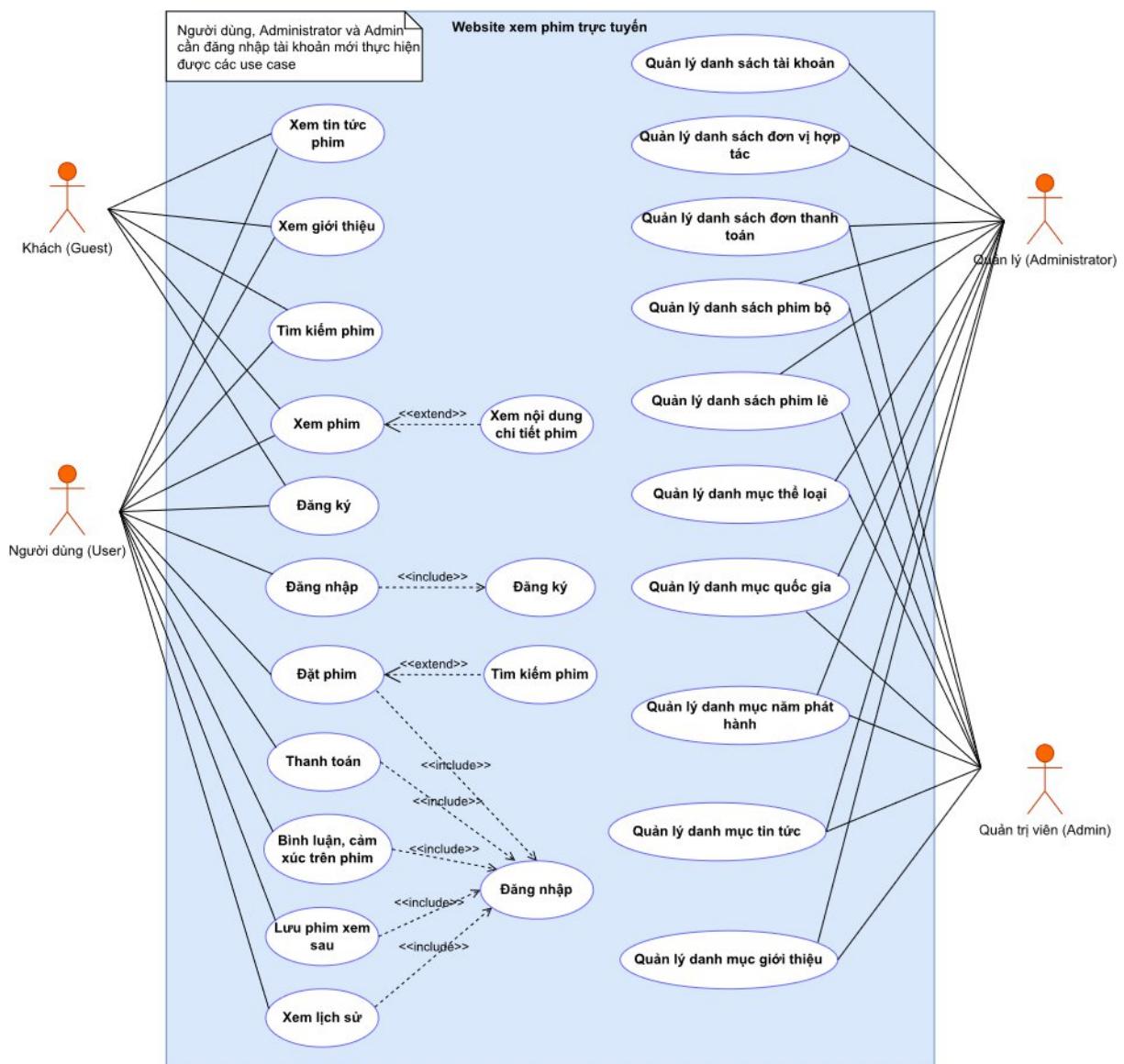
QĐ5 :

+ Tất cả dữ liệu khác được bỏ trống.

3. MÔ HÌNH HÓA YÊU CẦU

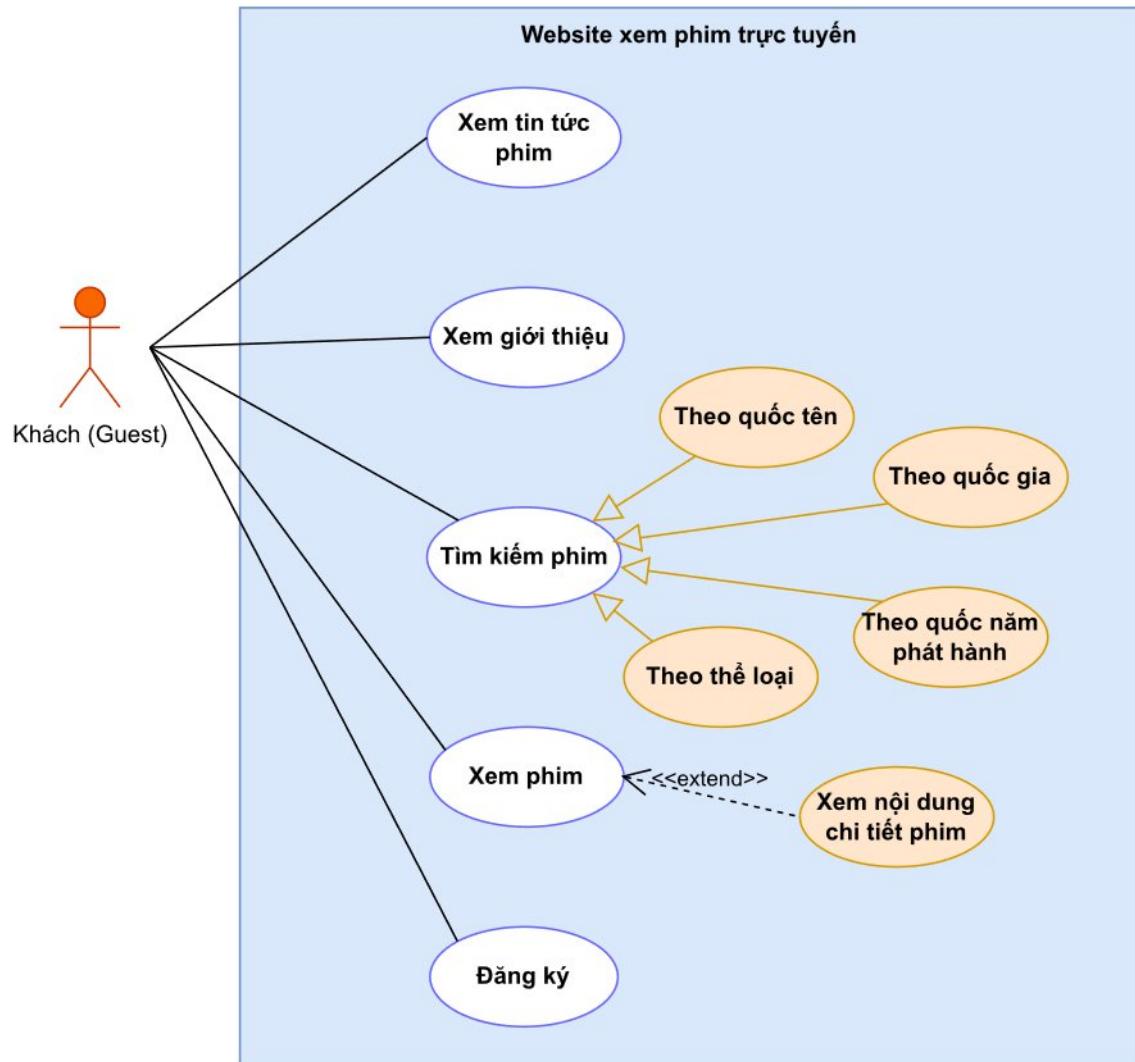
3.1. Usecase Diagram

3.1.1. Sơ đồ mức tổng quát



Hình 3-1 UseCase Diagram tổng quát

3.1.2. Sơ đồ use case Khách (Guest)



Hình 3-2 Sơ đồ use case Khách (Guest)

3.1.3. Sơ đồ use case Người dùng (User)



Hình 3-3 Sơ đồ use case Người dùng (User)

3.1.4. Sơ đồ use case Quản trị viên (Admin)



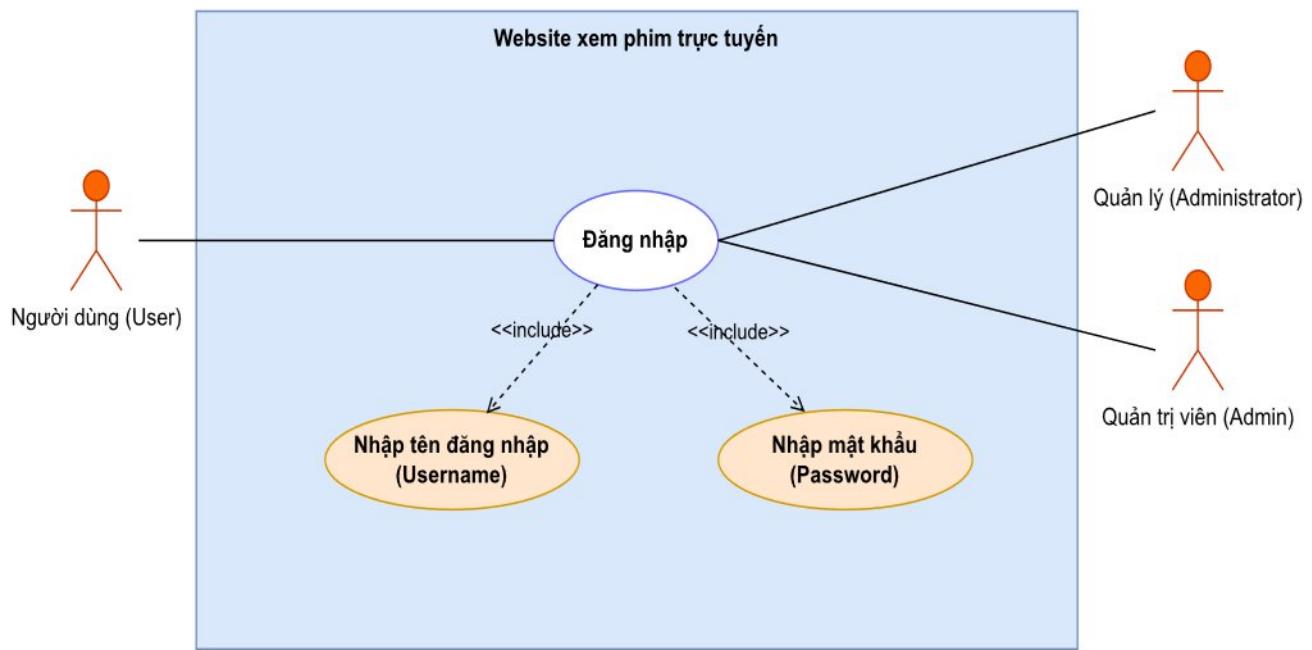
Hình 3-4 Sơ đồ use case Quản trị viên (Admin)

3.1.5. Sơ đồ use case Quản lý website (Administrator)



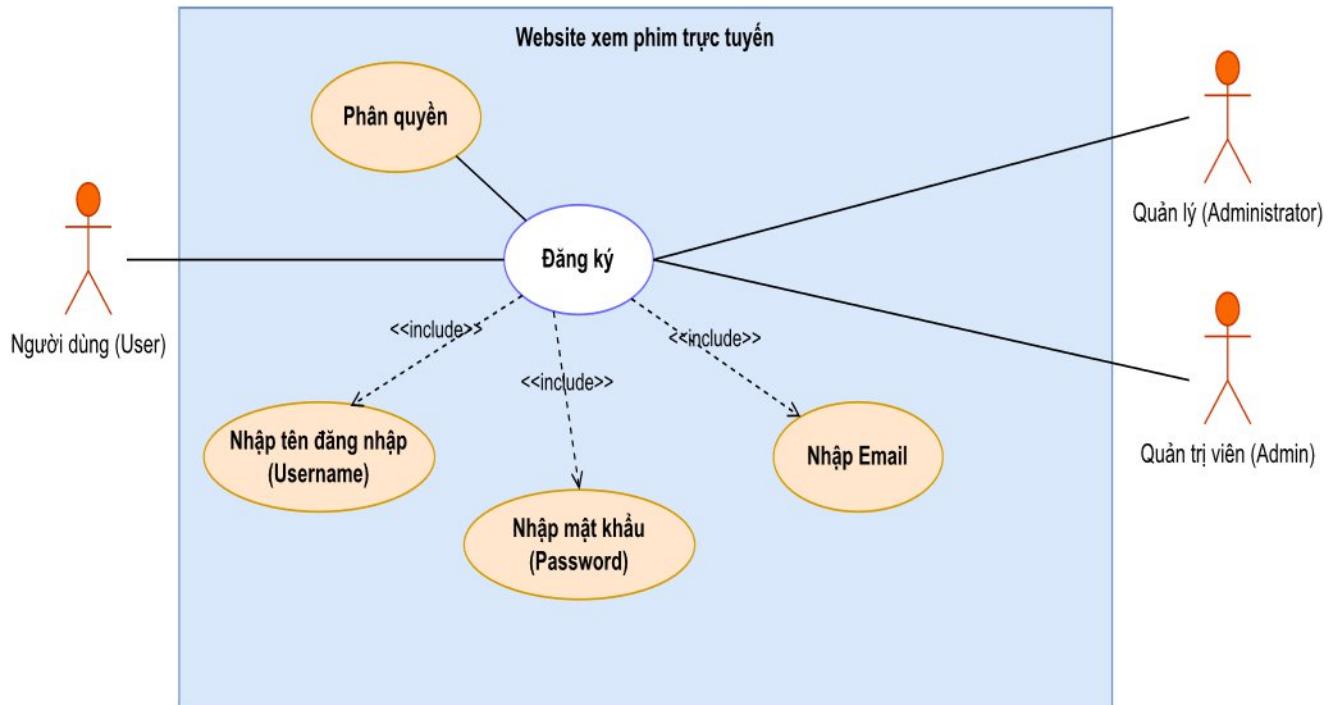
Hình 3-5 Sơ đồ use case Quản lý website (Administrator)

3.1.6. Sơ đồ use case Đăng nhập (Login)



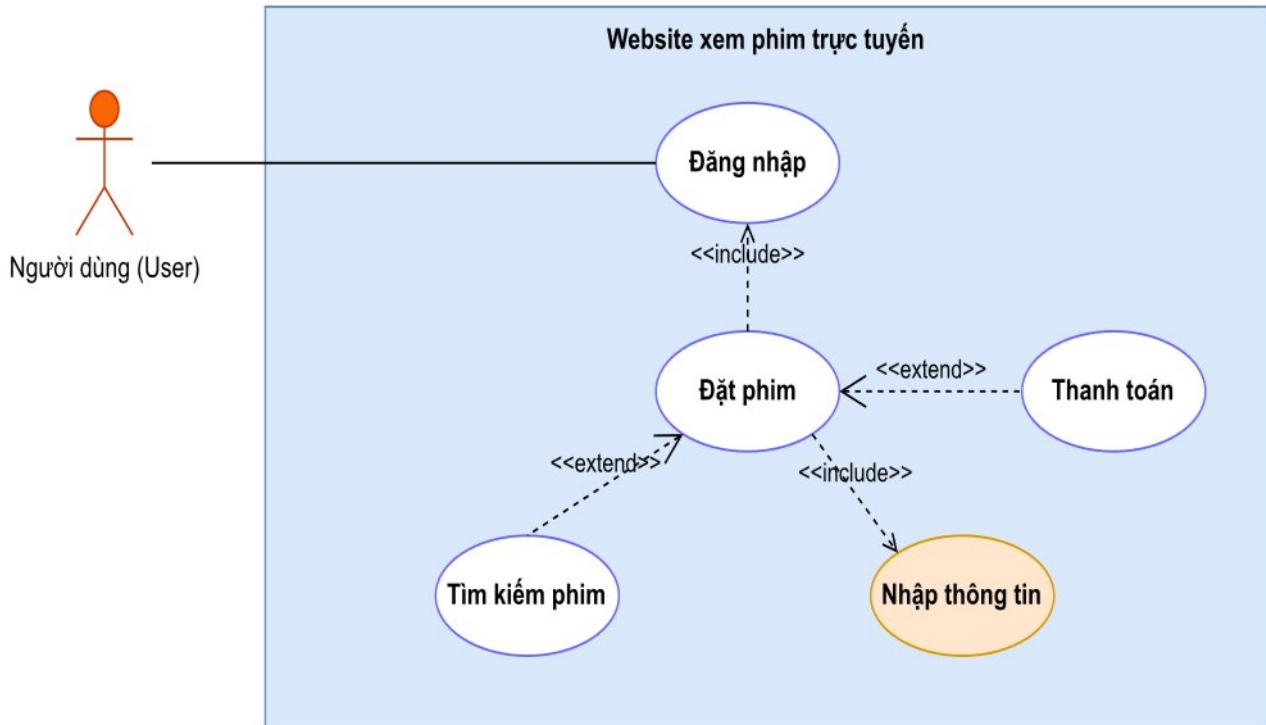
Hình 3-6 Sơ đồ use case Đăng nhập (Login)

3.1.7. Sơ đồ use case Đăng ký



Hình 3-7 Sơ đồ use case Đăng ký

3.1.8. Sơ đồ use case Đặt phim



Hình 3-8 Sơ đồ use case Đặt phim

3.2. Bảng Usecase

Bảng 3-1 Bảng Usecase

Code	Tên Usecase
UC01	Đăng ký tài khoản
UC02	Đăng nhập
UC03	Sửa thông tin cá nhân
UC04	Tìm kiếm phim
UC05	Xem phim
UC06	Thêm phim vào mục yêu thích
UC07	Xóa phim khỏi mục yêu thích
UC08	Bình luận, cảm xúc phim
UC09	Thêm phim
UC10	Xóa phim
UC11	Sửa chi tiết phim
UC12	Đặt mua phim
UC13	Xem chi tiết phim
UC14	Thêm tin tức
UC15	Thông kê

3.3. Đặc tả Usecase

3.3.1. Đăng ký tài khoản

Bảng 3-2 UC Đăng ký tài khoản

Use case ID	UC1
Tên use case	UC Đăng ký tài khoản
Mô tả	Use case này cho phép người dùng khách đăng ký tài khoản để trở thành người dùng chính
Actor chính	Khách (Guest)
Actor phụ	Không có
Tiền điều kiện	Không có
Hậu điều kiện	Người dùng khách sẽ trở thành người dùng chính của website xem phim.
Luồng hoạt động	<ol style="list-style-type: none">Người dùng khách chọn button có icon tài khoản , sau đó chọn chức năng “Đăng nhập” và chọn chức năng “ Đăng ký tài khoản”cHệ thống hiển thị form đăng kýNgười dùng khách nhập thông tin cần thiết vào form và nhấn nút “Đăng ký”Hệ thống thông báo kết quả nhập thông tinHệ thống cập nhật thông tin vào danh sách tài khoản
Luồng thay thế	<ol style="list-style-type: none">Nhập sai thông tin hoặc bỏ trống<ol style="list-style-type: none">Hệ thống thông báo nhập sai thông tin và yêu cầu người dùng khách nhập lại thông tinNếu người dùng khách đồng ý thì quay lại bước 2. Ngược lại, use case kết thúc.

3.3.2. Đăng nhập

Bảng 3-3 UC Đăng nhập

Use case ID	UC2
Tên use case	UC Đăng nhập
Mô tả	Use case này cho phép người dùng đăng nhập vào hệ thống
Actor chính	Người dùng: Người dùng chính, Admin, Administrator
Actor phụ	Không có
Tiền điều kiện	Tài khoản người dùng phải tồn tại
Hậu điều kiện	<ul style="list-style-type: none"> - Người dùng đăng nhập thành công vào hệ thống - Hệ thống chuyển hướng đến trang làm việc thuộc phân quyền của người dùng
Luồng hoạt động	<ol style="list-style-type: none"> 1. Người dùng chọn chức năng “Đăng nhập” 2. Hệ thống yêu cầu người dùng nhập thông tin đăng nhập bao gồm tên đăng nhập (username) và mật khẩu (password) 3. Người dùng nhập thông tin và nhấn nút “Đăng nhập” 4. Hệ thống kiểm tra thông tin đăng nhập 5. Hệ thống thông báo đăng nhập thành công
Luồng thay thế	<ol style="list-style-type: none"> 4.1. Nếu sai thông tin đăng nhập, hệ thống hiển thị thông báo lỗi <ol style="list-style-type: none"> 4.1.1. Hệ thống yêu cầu người dùng nhập lại. Quay về bước 3

3.3.3. Sửa thông tin cá nhân

Bảng 3-4 UC Sửa thông tin cá nhân

Use case ID	UC3
Tên use case	UC Sửa thông tin cá nhân
Mô tả	Use case này cho phép người dùng sửa đổi thông tin cá nhân của mình, bao gồm tên, email, mật khẩu, và các thông tin liên quan khác.
Actor chính	Người dùng: Người dùng đã đăng nhập vào hệ thống.
Actor phụ	Không có
Tiền điều kiện	Người dùng đã đăng nhập vào hệ thống
Hậu điều kiện	<ul style="list-style-type: none"> - Hệ thống cập nhật thành công thông tin cá nhân của người dùng - Người dùng có thể xem lại thông tin đã cập nhật trên giao diện.
Luồng hoạt động	<ol style="list-style-type: none"> 1. Hệ thống hiển thị giao diện thông tin cá nhân Hệ thống hiển thị thông tin cá nhân hiện tại của người dùng, bao gồm tên, email, mật khẩu (ẩn), và các thông tin cá nhân khác. 2. Người dùng chỉnh sửa thông tin cá nhân Người dùng có thể chỉnh sửa các thông tin sau:<ul style="list-style-type: none"> - Tên - Email - Mật khẩu - Số điện thoại đăng ký 3. Người dùng nhấn nút "Cập nhật" Sau khi người dùng chỉnh sửa các thông tin, họ nhấn nút "Cập nhật" để hệ thống xử lý yêu cầu. 4. Hệ thống kiểm tra và xác thực thông tin Hệ thống kiểm tra tính hợp lệ của các thông tin mà người dùng nhập vào:<ul style="list-style-type: none"> - Kiểm tra xem email có đúng định dạng hay không. - Kiểm tra độ dài và tính hợp lệ của mật khẩu (nếu có thay đổi mật khẩu).

	<ul style="list-style-type: none"> - Kiểm tra xem tên người dùng đã tồn tại trong hệ thống hay chưa (nếu có thay đổi). <p>5. Cập nhật thông tin và phản hồi cho người dùng</p> <ul style="list-style-type: none"> - Nếu thông tin hợp lệ, hệ thống sẽ lưu các thay đổi vào cơ sở dữ liệu và hiển thị thông báo "Cập nhật thông tin thành công". - Nếu có lỗi (như email không hợp lệ, tên người dùng đã tồn tại, mật khẩu không đủ mạnh), hệ thống sẽ thông báo lỗi và yêu cầu người dùng sửa lại thông tin.
Luồng thay thế	<p>3.1. Người dùng không chỉnh sửa thông tin:</p> <p>3.1.1. Người dùng có thể chọn "Quay lại" mà không chỉnh sửa bất kỳ thông tin nào, và hệ thống sẽ đưa họ quay lại trang thông tin cá nhân mà không thay đổi gì.</p> <p>4.1. Kiểm tra thông tin không hợp lệ:</p> <p>4.1.1. Nếu người dùng nhập thông tin không hợp lệ (ví dụ: email sai định dạng, mật khẩu quá yếu), hệ thống sẽ thông báo lỗi và yêu cầu người dùng chỉnh sửa lại thông tin.</p> <p>4.1.2. Người dùng sẽ trở lại bước 2 và nhập lại thông tin hợp lệ.</p> <p>4.2. Tên người dùng đã tồn tại:</p> <p>4.2.1. Nếu người dùng thay đổi tên đăng nhập (hoặc thông tin quan trọng khác) và thông tin đó đã tồn tại trong hệ thống, hệ thống sẽ thông báo lỗi và yêu cầu người dùng chọn tên mới</p>

3.3.4. Tìm kiếm phim

Bảng 3-5 UC Tìm kiếm phim

Use case ID	UC4
Tên use case	UC Tìm kiếm phim
Mô tả	Use case này cho phép người dùng tìm kiếm phim theo nhiều tiêu chí khác nhau
Actor chính	Người dùng: Người dùng chính, người dùng khách (Guest), Admin, Administrator
Actor phụ	Không có
Tiền điều kiện	Người dùng truy cập vào website
Hậu điều kiện	<ul style="list-style-type: none"> - Hệ thống hiển thị ra màn hình danh sách phim cần tìm - Người dùng có thể xem thông tin chi tiết của phim
Luồng hoạt động	<ol style="list-style-type: none"> 1. Hệ thống hiển thị ra màn hình menu tiêu chí dạng dropdown list danh sách và ô tìm kiếm để thực hiện tìm kiếm phim: <ul style="list-style-type: none"> - Tìm kiếm theo tên phim (Nhập tên phim vào ô tìm kiếm) - Tìm kiếm theo loại phim (Phim bộ & phim lẻ) - Tìm kiếm theo thể loại phim - Tìm kiếm theo năm phát hành phim - Tìm kiếm theo quốc gia sản xuất phim 2. Người dùng chọn các tiêu chí theo nhu cầu và nhập các giá trị cần thiết. 3. Người dùng nhấn nút “Enter” hoặc Click chuột vào menu dropdown list tương ứng. 4. Hệ thống lọc danh mục phim, các phim tương ứng và hiển thị ra màn hình
Luồng thay thế	3.1. Không chọn tiêu chí nào

	<p>3.1.1. Mặc định sẽ hiển thị trang chủ liệt kê danh sách các phim bộ và phim lẻ</p> <p>4.1. Không có tên phim nào phù hợp tiêu chí</p> <p> 4.1.1. Hệ thống thông báo “Không có phim thích hợp”</p> <p> Quay lại bước 1, nếu người dùng muốn chọn lại</p>
--	--

3.3.5. Xem phim

Bảng 3-6 UC Xem phim

Use case ID	UC5
Tên use case	UC Xem phim
Mô tả	Use case này cho phép người dùng xem phim, bao gồm việc chọn phim từ danh sách, xem thông tin chi tiết về phim, và xem nội dung phim (video hoặc trailer).
Actor chính	Người dùng: Người dùng chính, Admin, Administrator
Actor phụ	Không có
Tiền điều kiện	<ul style="list-style-type: none"> - Người dùng đã đăng nhập vào hệ thống - Người dùng đã truy cập vào trang chi tiết phim hoặc trang danh sách phim. - Phim đã được tải lên và có thể phát trên hệ thống.
Hậu điều kiện	Người dùng có thể thoát khỏi trang phim hoặc quay lại trang danh sách phim.
Luồng hoạt động	<ol style="list-style-type: none"> 1. Hệ thống hiển thị danh sách phim Người dùng có thể duyệt qua các phim theo nhiều tiêu chí khác nhau như thể loại, năm phát hành, độ phổ biến 2. Người dùng chọn phim Người dùng chọn một bộ phim từ danh sách để xem. Hệ thống sẽ chuyển

	<p>hướng đến trang chi tiết của bộ phim đó.</p> <p>3. Hệ thống hiển thị thông tin chi tiết về phim</p> <p>Hệ thống hiển thị thông tin chi tiết về bộ phim, bao gồm:</p> <ul style="list-style-type: none"> - Tên phim - Thể loại - Quốc gia sản xuất - Năm phát hành - Nội dung phim tóm tắt - Đánh giá, lượt xem. <p>4. Người dùng nhấn nút "Xem Phim"</p> <p>Người dùng nhấn nút "Play" để bắt đầu xem phim. Phim sẽ được phát trực tiếp trên giao diện hoặc thông qua một player video.</p> <p>5. Người dùng xem phim</p> <p>Người dùng xem phim với các tính năng điều khiển như:</p> <ul style="list-style-type: none"> - Play/Pause - Âm lượng - Chuyển động đến thời gian khác trong phim - Chế độ toàn màn h <p>6. Người dùng có thể dừng xem và quay lại</p> <p>Khi người dùng muốn ngừng xem, họ có thể nhấn nút "Dừng" hoặc quay lại trang trước đó để xem phim khác.</p>
Luồng thay thế	<p>6.1. Phim không có sẵn để xem:</p> <p>6.1.1. Nếu phim không có sẵn để xem hoặc người dùng không có quyền truy cập (ví dụ: chỉ dành cho người dùng VIP), hệ thống sẽ thông báo "Phim không có sẵn cho bạn. Vui lòng nâng cấp tài khoản."</p>

3.3.6. Thêm phim vào mục yêu thích

Bảng 3-7 UC Thêm phim vào mục yêu thích

Use case ID	UC6
Tên use case	UC Thêm phim vào mục yêu thích
Mô tả	Use case này cho phép người dùng thêm các bộ phim vào mục yêu thích của họ để có thể dễ dàng truy cập lại các phim yêu thích trong tương lai.
Actor chính	Người dùng: Người dùng chính, Admin, Administrator
Actor phụ	Không có
Tiền điều kiện	<ul style="list-style-type: none"> - Người dùng đã đăng nhập vào hệ thống - Người dùng đã duyệt qua danh sách phim và muốn thêm một bộ phim vào mục yêu thích.
Hậu điều kiện	<ul style="list-style-type: none"> - Bộ phim được thêm vào danh sách yêu thích của người dùng. - Người dùng có thể truy cập vào danh sách yêu thích bất cứ khi nào.
Luồng hoạt động	<ol style="list-style-type: none"> 1. Hệ thống hiển thị danh sách phim Hệ thống hiển thị danh sách phim cho người dùng chọn, bao gồm các bộ phim yêu thích và phim chưa yêu thích. 2. Người dùng chọn bộ phim để thêm vào yêu thích Người dùng nhấn nút "Thêm vào yêu thích" (hoặc biểu tượng trái tim) trên phim mà họ muốn thêm vào danh sách yêu thích. 3. Hệ thống kiểm tra trạng thái yêu thích của phim Hệ thống sẽ kiểm tra xem phim này đã có trong danh sách yêu thích của người dùng hay chưa: <ul style="list-style-type: none"> - Nếu phim đã có trong danh sách yêu thích, hệ thống sẽ thông báo "Phim này đã có trong danh sách yêu thích". - Nếu phim chưa có trong danh sách yêu thích, hệ thống sẽ thêm

	<p>phim vào danh sách yêu thích của người dùng.</p> <p>4. Hệ thống xác nhận việc thêm phim vào yêu thích</p> <p>Sau khi thêm phim vào danh sách yêu thích, hệ thống sẽ thông báo "Phim đã được thêm vào danh sách yêu thích" cho người dùng.</p> <p>5. Người dùng có thể xem lại danh sách yêu thích</p> <p>Người dùng có thể truy cập vào phần "Danh sách yêu thích" của mình để xem tất cả các bộ phim đã được thêm vào yêu thích.</p>
Luồng thay thế	<p>3.1. Người dùng đã thêm phim vào yêu thích trước đó:</p> <p>3.1.1. Nếu phim đã được thêm vào yêu thích từ trước, hệ thống sẽ thông báo "Phim này đã có trong danh sách yêu thích" mà không thực hiện thay đổi gì.</p> <p>4.1. Phim không thể thêm vào yêu thích:</p> <p>4.1.1. Nếu có lỗi xảy ra khi thêm phim vào yêu thích (ví dụ: lỗi kết nối cơ sở dữ liệu), hệ thống sẽ thông báo "Không thể thêm phim vào yêu thích. Vui lòng thử lại sau."</p> <p>5.1 Người dùng không có quyền truy cập danh sách yêu thích:</p> <p>5.1.1. Nếu người dùng không có quyền truy cập vào danh sách yêu thích (ví dụ: tài khoản của họ bị khóa), hệ thống sẽ thông báo "Bạn không có quyền truy cập vào danh sách yêu thích.".</p>

3.3.7. Xóa phim khỏi mục yêu thích

Bảng 3-8 UC Xóa phim khỏi mục yêu thích

Use case ID	UC7
Tên use case	UC Xóa phim khỏi mục yêu thích
Mô tả	Use case này cho phép người dùng xóa các bộ phim đã thêm vào mục yêu thích của họ, giúp người dùng quản lý lại danh sách phim yêu thích của

	mình.
Actor chính	Người dùng: Người dùng chính, Admin, Administrator
Actor phụ	Không có
Tiền điều kiện	<ul style="list-style-type: none"> - Người dùng đã đăng nhập vào hệ thống - Người dùng có ít nhất một bộ phim trong danh sách yêu thích.
Hậu điều kiện	<ul style="list-style-type: none"> - Bộ phim được xóa khỏi danh sách yêu thích của người dùng. - Người dùng không còn thấy bộ phim đó trong danh sách yêu thích khi truy cập vào mục yêu thích.
Luồng hoạt động	<ol style="list-style-type: none"> 1. Hệ thống hiển thị danh sách yêu thích của người dùng Người dùng truy cập vào mục "Yêu thích" trong tài khoản của mình để xem danh sách các bộ phim đã thêm vào yêu thích. 2. Người dùng chọn bộ phim để xóa khỏi yêu thích Trong danh sách yêu thích, người dùng nhấn nút "Xóa" (hoặc biểu tượng thùng rác) trên bộ phim mà họ muốn xóa khỏi danh sách yêu thích. 3. Hệ thống xác nhận yêu cầu xóa phim Hệ thống yêu cầu người dùng xác nhận hành động xóa bộ phim khỏi danh sách yêu thích: <ul style="list-style-type: none"> - Người dùng có thể chọn "Xóa" để xác nhận hoặc "Hủy" nếu họ không muốn xóa phim. 4. Hệ thống xóa phim khỏi danh sách yêu thích Sau khi người dùng xác nhận xóa, hệ thống sẽ loại bỏ bộ phim khỏi danh sách yêu thích của người dùng. 5. Hệ thống thông báo xóa phim thành công Sau khi xóa phim thành công, hệ thống sẽ thông báo "Phim đã được xóa khỏi danh sách yêu thích". 6. Người dùng quay lại danh sách yêu thích Người dùng có thể quay lại danh sách yêu thích để xác nhận bộ phim đã

	được xóa khỏi danh sách.
Luồng thay thế	<p>3.1. Người dùng không xác nhận xóa phim:</p> <p>3.1.1. Nếu người dùng chọn "Hủy", hệ thống sẽ không xóa bộ phim và người dùng quay lại trang danh sách yêu thích mà không thay đổi gì.</p> <p>4.1. Lỗi khi xóa phim:</p> <p>4.1.1. Nếu có lỗi xảy ra trong quá trình xóa (ví dụ: sự cố với cơ sở dữ liệu), hệ thống sẽ thông báo "Không thể xóa phim khỏi danh sách yêu thích. Vui lòng thử lại sau."</p> <p>5.1. Phim không có trong danh sách yêu thích:</p> <p>5.1.1. Nếu người dùng cố gắng xóa phim mà không có trong danh sách yêu thích (do đã bị xóa trước đó hoặc lỗi), hệ thống sẽ thông báo "Phim không có trong danh sách yêu thích".</p>

3.3.8. Bình luận, cảm xúc phim

Bảng 3-9 UC Bình luận, cảm xúc phim

Use case ID	UC8
Tên use case	UC Bình luận, cảm xúc phim
Mô tả	Use case này cho phép người dùng có thể để lại biểu đạt cảm xúc yêu thích, ý kiến cá nhân bên dưới phần bình luận của phim đang xem
Actor chính	Người dùng: Người dùng chính, Admin, Administrator
Actor phụ	Không có
Tiền điều kiện	Người dùng đã đăng nhập vào hệ thống, đang xem một phim cụ thể,

Hậu điều kiện	Không có
Luồng hoạt động	<p>1. Người dùng chọn nút “Bình luận” bên dưới khung video xem phim, hoặc rê chuột vào nút yêu thích để thả cảm xúc cho phim</p> <p>2. Hệ thống hiển thị ra màn hình form bình luận (chọn nút “Bình luận”)</p> <p>3. Người dùng viết bình luận, có thể đính kèm ảnh và nhấn nút “Gửi”</p> <p>4. Hệ thống lưu lại bình luận vào cơ sở dữ liệu và hiển thị lên màn hình</p>
Luồng thay thế	<p>3.1. Nội dung bình luận trống nhưng người dùng bấm “Gửi” sẽ xuất ra thông báo yêu cầu nhập nội dung bình luận trước khi gửi</p>

3.3.9. Thêm phim

Bảng 3-10 UC Thêm phim

Use case ID	UC9
Tên use case	UC Thêm phim
Mô tả	Use case này cho phép Quản trị viên và Quản lý website thêm tin tức phim mới
Actor chính	Người quản trị: Admin và Administrator
Actor phụ	Không có
Tiền điều kiện	Người quản trị đã đăng nhập vào hệ thống
Hậu điều kiện	Hệ thống thông báo cập nhật thành công
Luồng hoạt động	<ol style="list-style-type: none"> 1. Hệ thống hiển thị giao diện soạn thảo tin tức và yêu cầu người quản trị nhập thông tin 2. Người quản trị nhập phim bộ và tập phim của phim bộ và nhấn nút “Thêm” 3. Hệ thống kiểm tra và thêm tin tức vào cơ sở dữ liệu 4. Hệ thống cập nhật lại danh sách phim bộ và hiển thị lên màn hình
Luồng thay thế	<ol style="list-style-type: none"> 3.1. Phim thêm có tên trùng với phim đã có thì báo lỗi trùng phim và yêu cầu nhập lại. Quay lại bước 1

3.3.10.Xóa phim

Bảng 3-11 UC Xóa phim

Use case ID	UC10
Tên use case	UC Xóa phim
Mô tả	Use case này cho phép người dùng hoặc quản trị viên xóa một bộ phim khỏi hệ thống, ví dụ như khi phim đã hết hạn, không còn được phát hành, hoặc không còn cần thiết.
Actor chính	Người quản trị: Admin và Administrator

Actor phụ	Không có
Tiền điều kiện	<ul style="list-style-type: none"> - Người quản trị (hoặc người dùng có quyền) đã đăng nhập vào hệ thống và có quyền xóa phim. - Phim đã có trên hệ thống và có thể xóa.
Hậu điều kiện	<ul style="list-style-type: none"> - Phim sẽ bị xóa khỏi hệ thống và không còn hiển thị trong danh sách phim. - Tất cả các dữ liệu liên quan đến phim (như video, thông tin mô tả, hình ảnh) sẽ bị xóa.
Luồng hoạt động	<p>Hệ thống hiển thị danh sách các bộ phim</p> <p>Hệ thống hiển thị danh sách các bộ phim mà người quản trị (hoặc người dùng có quyền) có thể quản lý và xóa.</p> <p>2. Người quản trị chọn bộ phim muốn xóa</p> <p>Người quản trị chọn một bộ phim từ danh sách phim để xóa. Phim có thể là phim chưa hoàn thành, phim không còn hợp lệ hoặc phim cần được xóa vì lý do khác.</p> <p>3. Hệ thống yêu cầu xác nhận xóa phim</p> <p>Hệ thống yêu cầu người quản trị xác nhận hành động xóa phim. Người quản trị sẽ nhận được thông báo như: "Bạn có chắc chắn muốn xóa phim này? Tất cả dữ liệu liên quan sẽ bị mất."</p> <p>4. Người quản trị xác nhận xóa phim</p> <p>Người quản trị nhấn "Xác nhận" để tiếp tục xóa phim hoặc chọn "Hủy" để bỏ qua hành động này.</p> <p>5. Hệ thống xóa phim khỏi hệ thống</p> <p>Nếu người quản trị xác nhận, hệ thống sẽ xóa phim khỏi cơ sở dữ liệu và tất cả các thông tin liên quan (video, ảnh, mô tả) sẽ bị xóa.</p> <p>6. Hệ thống thông báo xóa phim thành công</p> <p>Sau khi xóa phim thành công, hệ thống sẽ hiển thị thông báo "Phim đã được xóa thành công".</p>

	<p>7. Người quản trị quay lại danh sách phim</p> <p>Người quản trị có thể quay lại danh sách phim để tiếp tục quản lý các bộ phim còn lại.</p>
Luồng thay thế	<p>3.1. Người quản trị không xác nhận xóa phim:</p> <p>3.1.1. Nếu người quản trị chọn "Hủy", hệ thống sẽ không xóa phim và quay lại danh sách phim mà không thay đổi gì.</p> <p>4.1. Lỗi khi xóa phim:</p> <p>4.1.1. Nếu có lỗi xảy ra trong quá trình xóa (ví dụ: lỗi kết nối cơ sở dữ liệu), hệ thống sẽ thông báo "Không thể xóa phim. Vui lòng thử lại sau."</p> <p>5.1. Phim đã bị xóa từ trước:</p> <p>5.1.1. Nếu người quản trị cố gắng xóa phim đã bị xóa trước đó hoặc không tồn tại trong hệ thống, hệ thống sẽ thông báo "Phim không tồn tại hoặc đã bị xóa."</p>

3.3.11. Sửa chi tiết phim

Bảng 3-12 UC Sửa chi tiết phim.

Use case ID	UC11
Tên use case	UC Sửa chi tiết phim.
Mô tả	Use case này cho phép người quản trị (hoặc người có quyền quản lý phim) sửa thông tin chi tiết của một bộ phim đã có trong hệ thống, bao gồm tên phim, thể loại, năm phát hành, mô tả, và các thông tin khác.
Actor chính	Người dùng: Admin, Administrator
Actor phụ	Không có
Tiền điều kiện	<ul style="list-style-type: none"> - Người dùng đã đăng nhập vào hệ thống - Bộ phim đã có trong hệ thống và có thể chỉnh sửa thông tin.
Hậu điều kiện	<ul style="list-style-type: none"> - Các thông tin phim đã được cập nhật thành công trong hệ thống. - Bộ phim sẽ hiển thị các thông tin mới khi người dùng truy cập vào chi

	tiết phim.
Luồng hoạt động	<p>1. Hệ thống hiển thị danh sách phim</p> <p>Hệ thống hiển thị danh sách các bộ phim mà người quản trị (hoặc người có quyền) có thể chỉnh sửa.</p> <p>2. Người quản trị chọn bộ phim cần sửa</p> <p>Người quản trị chọn bộ phim mà họ muốn sửa thông tin. Sau khi chọn, hệ thống sẽ chuyển đến trang chỉnh sửa thông tin chi tiết của bộ phim.</p> <p>3. Hệ thống hiển thị form chỉnh sửa thông tin phim</p> <p>Hệ thống hiển thị các trường thông tin phim mà người quản trị có thể chỉnh sửa, bao gồm:</p> <ul style="list-style-type: none"> - Tên Phim - Thể loại - Quốc gia sản xuất - Diễn Viên - Nội dung phim tóm tắt - Hình bìa phim <p>4. Người quản trị sửa thông tin phim</p> <p>Người quản trị thay đổi các thông tin phim theo nhu cầu, ví dụ như thay đổi tên phim, cập nhật thông tin về diễn viên, năm phát hành, v.v.</p> <p>5. Người quản trị lưu thay đổi</p> <p>Sau khi hoàn tất việc sửa, người quản trị nhấn nút "Lưu thay đổi" để lưu các thông tin đã chỉnh sửa.</p> <p>6. Hệ thống xác nhận thay đổi</p> <p>Hệ thống xác nhận rằng các thông tin đã được cập nhật thành công và thông báo "Thông tin phim đã được cập nhật".</p> <p>7. Người quản trị quay lại danh sách phim</p>

	Người quản trị có thể quay lại danh sách phim để tiếp tục quản lý các bộ phim khác hoặc kiểm tra bộ phim đã sửa.
Luồng thay thế	<p>4.1. Lỗi khi lưu thông tin phim:</p> <p>4.1.1. Nếu có lỗi xảy ra trong quá trình lưu thông tin phim (ví dụ: lỗi kết nối cơ sở dữ liệu), hệ thống sẽ thông báo "Không thể lưu thay đổi. Vui lòng thử lại sau."</p> <p>5.1. Dữ liệu không hợp lệ:</p> <p>5.1.1. Nếu người quản trị nhập thông tin không hợp lệ (ví dụ: tên phim trống, năm phát hành không hợp lệ), hệ thống sẽ hiển thị thông báo lỗi tương ứng, chẳng hạn như "Tên phim không được để trống" hoặc "Năm phát hành không hợp lệ".</p> <p>6.1. Hủy bỏ thay đổi:</p> <p>6.1.1. Nếu người quản trị chọn hủy bỏ thay đổi trước khi lưu, hệ thống sẽ không lưu lại bất kỳ thay đổi nào và quay lại màn hình chi tiết phim ban đầu.</p>

3.3.12. Đặt mua phim

Bảng 3-13 UC Đặt mua phim

Use case ID	UC12
Tên use case	UC Đặt mua phim
Mô tả	Use case này cho phép khách hàng đặt mua phim trực tuyến và thanh toán trực tuyến
Actor chính	Người dùng: Người dùng chính, Admin, Administrator
Actor phụ	Không có
Tiền điều kiện	- Người dùng đã đăng nhập vào hệ thống
Hậu điều kiện	Hệ thống thông báo đặt mua phim thành công và chuyển đến trang xuất

	thông tin hóa đơn cần thanh toán
Luồng hoạt động	<ol style="list-style-type: none"> 1. Người dùng chọn phim cần mua, chọn chức năng “Đặt mua phim” 2. Hệ thống hiển thị form yêu cầu nhập thông tin khách hàng và chọn hình thức thanh toán (Qua chuyển khoản ngân hàng hoặc các ví điện tử online) 3. Người dùng nhập thông tin vào form 4. Hệ thống kiểm tra dữ liệu nhập vào 5. Người dùng nhấn nút “Đặt mua phim” 6. Hệ thống lưu lại thông tin đặt mua phim và thông tin hóa đơn cần thanh toán của người dùng
Luồng thay thế	<ol style="list-style-type: none"> 4.1. Thời điểm thanh toán hóa đơn quá 48h (kể từ thời điểm đặt mua phim) <ol style="list-style-type: none"> 4.1.1. Hệ thống sẽ hủy hóa đơn đặt mua phim của người dùng đó. Xóa hóa đơn mua phim đó ra khỏi giỏ hàng của người dùng 4.2. Dữ liệu nhập sai thì hệ thống yêu cầu nhập lại.

3.3.13.Xem chi tiết phim

Bảng 3-14 UC Xem chi tiết phim

Use case ID	UC13
Tên use case	UC Xem chi tiết phim
Mô tả	Use case này cho phép người dùng xem thông tin chi tiết về nội dung phim (Nội dung, thể loại, thời lượng, năm phát hành ...)
Actor chính	Người dùng: Người dùng chính, người dùng khách (Guest), Admin, Administrator
Actor phụ	Không có

Tiền điều kiện	Người dùng truy cập vào website, có thể sử dụng use case “Tìm kiếm phim”
Hậu điều kiện	Người dùng có thể xem được thông tin phim và đưa ra lựa chọn xem video phim, xem sau hoặc đặt mua phim để xem
Luồng hoạt động	<p>1. Người dùng chọn vào hình đại diện của phim cần xem</p> <p>2. Hệ thống lấy thông tin chi tiết của phim được chọn trong cơ sở dữ liệu, bao gồm thông tin mã phim, tên phim, nội dung, năm phát hành, quốc gia, thời lượng ...</p> <p>3. Hệ thống hiển thị tất cả thông tin của phim tương ứng ra màn hình.</p>
Luồng thay thế	Không có

3.3.14.Thêm tin tức

Bảng 3-15 UC Thêm tin tức

Use case ID	UC14
Tên use case	UC Thêm tin tức
Mô tả	Use case này cho phép Quản trị viên và Quản lý website thêm tin tức phim mới
Actor chính	Người quản trị: Admin và Administrator
Actor phụ	Không có
Tiền điều kiện	Người quản trị đã đăng nhập vào hệ thống
Hậu điều kiện	Hệ thống thông báo cập nhật thành công
Luồng hoạt động	<ol style="list-style-type: none"> 1. Hệ thống hiển thị giao diện soạn thảo tin tức và yêu cầu người quản trị nhập thông tin 2. Người quản trị nhập tin tức và nhấn nút “Lưu” 3. Hệ thống kiểm tra và thêm tin tức vào cơ sở dữ liệu 4. Hệ thống cập nhật lại danh sách tin tức và hiển thị lên màn hình
Luồng thay thế	Không có

3.3.15.Thông kê

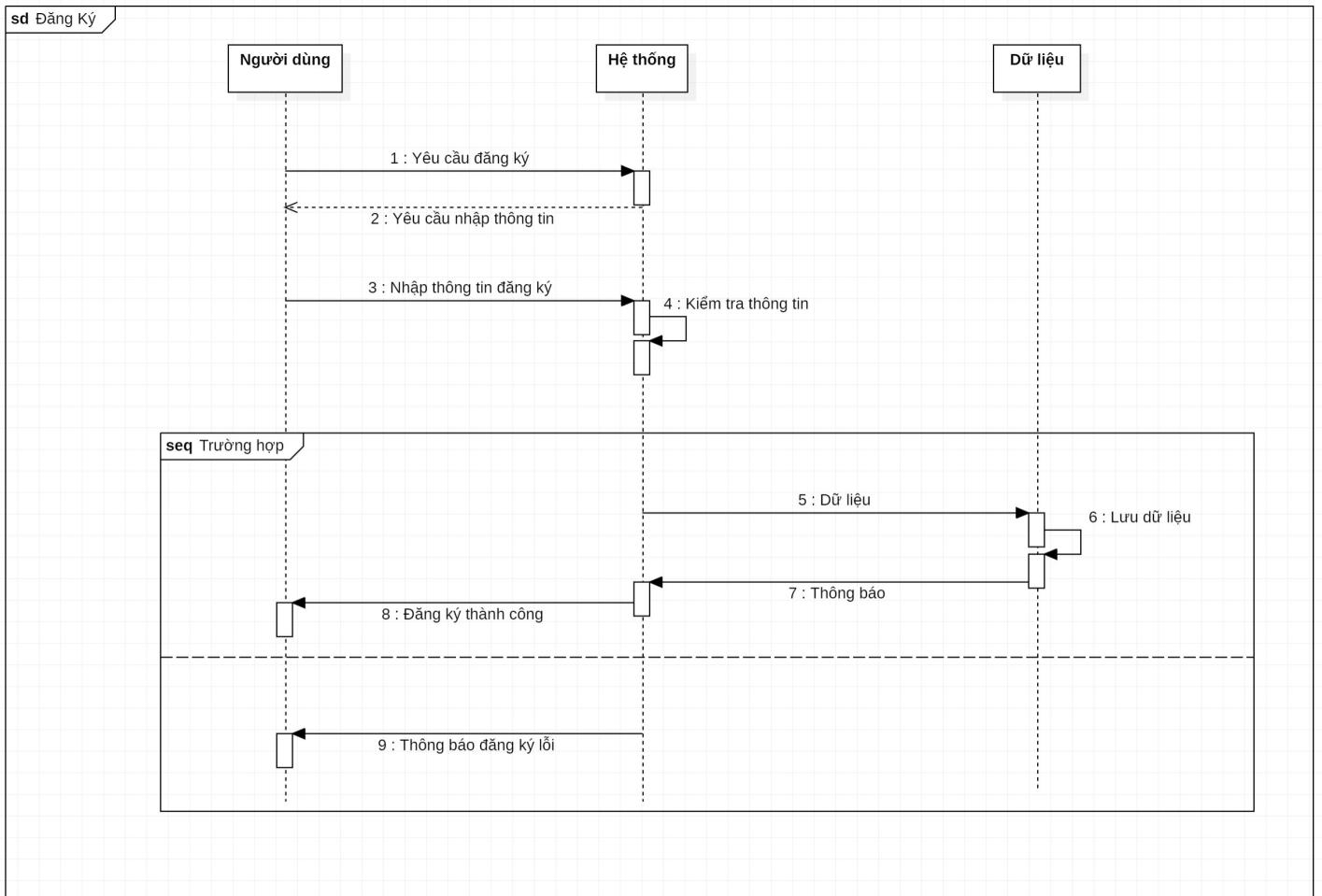
Bảng 3-16 UC Thông kê

Use case ID	UC15
Tên use case	UC Thông kê
Mô tả	Use case này cho phép người quản trị xem thông tin: <ul style="list-style-type: none"> - Thông kê số lượng tài khoản - Thông kê lượt xem của từng phim

	<ul style="list-style-type: none"> - Thống kê số lượt yêu thích của phim - Thống kê số lượng phim được mua nhiều nhất (Sắp xếp từ cao đến thấp)
Actor chính	Người quản trị: Admin và Administrator
Actor phụ	Máy in
Tiền điều kiện	Người quản trị đã đăng nhập vào hệ thống với quyền admin
Hậu điều kiện	Hệ thống hiển thị ra màn hình thống kê theo dạng bảng. Người quản trị có thể in báo cáo
Luồng hoạt động	<ol style="list-style-type: none"> 1. Tại giao diện trang chủ của người quản trị, hệ thống hiển thị các thông tin thống kê: <ul style="list-style-type: none"> - Thống kê số lượng tài khoản - Thống kê lượt xem của từng phim - Thống kê số lượt yêu thích của phim - Thống kê số lượng phim được mua nhiều nhất (Sắp xếp từ cao đến thấp) 2. Người quản trị nhấn nút “Xuất báo cáo” 3. Hệ thống sẽ hiển thị loại báo cáo, người quản trị chọn báo cáo cần xuất. Sau đó hệ thống sẽ xuất ra file in mẫu báo cáo có nội dung mà người quản trị chọn
Luồng thay thế	Không có

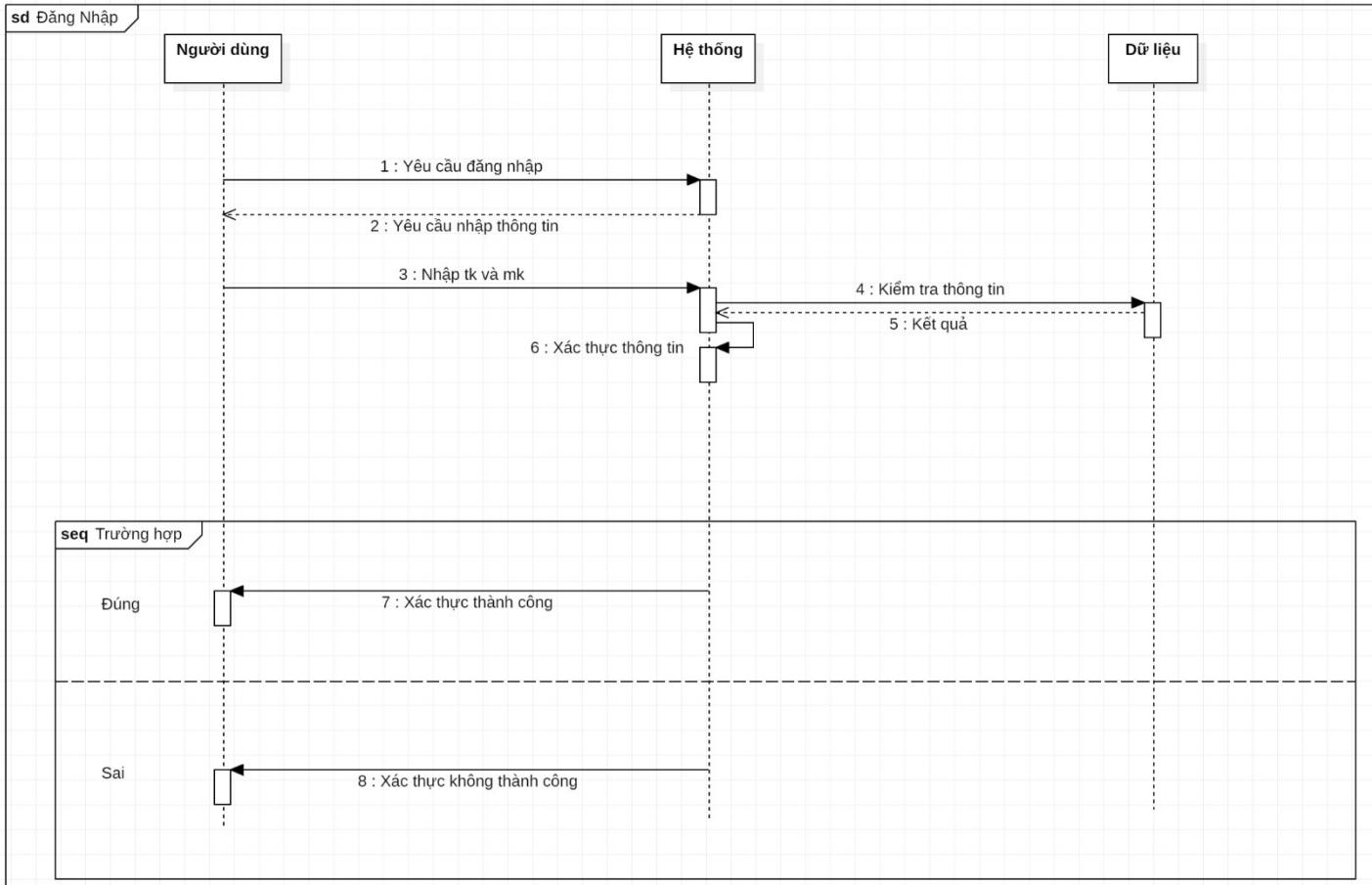
1.1 Sequence Diagram

1.1.1 Quy trình đăng ký tài khoản người dùng



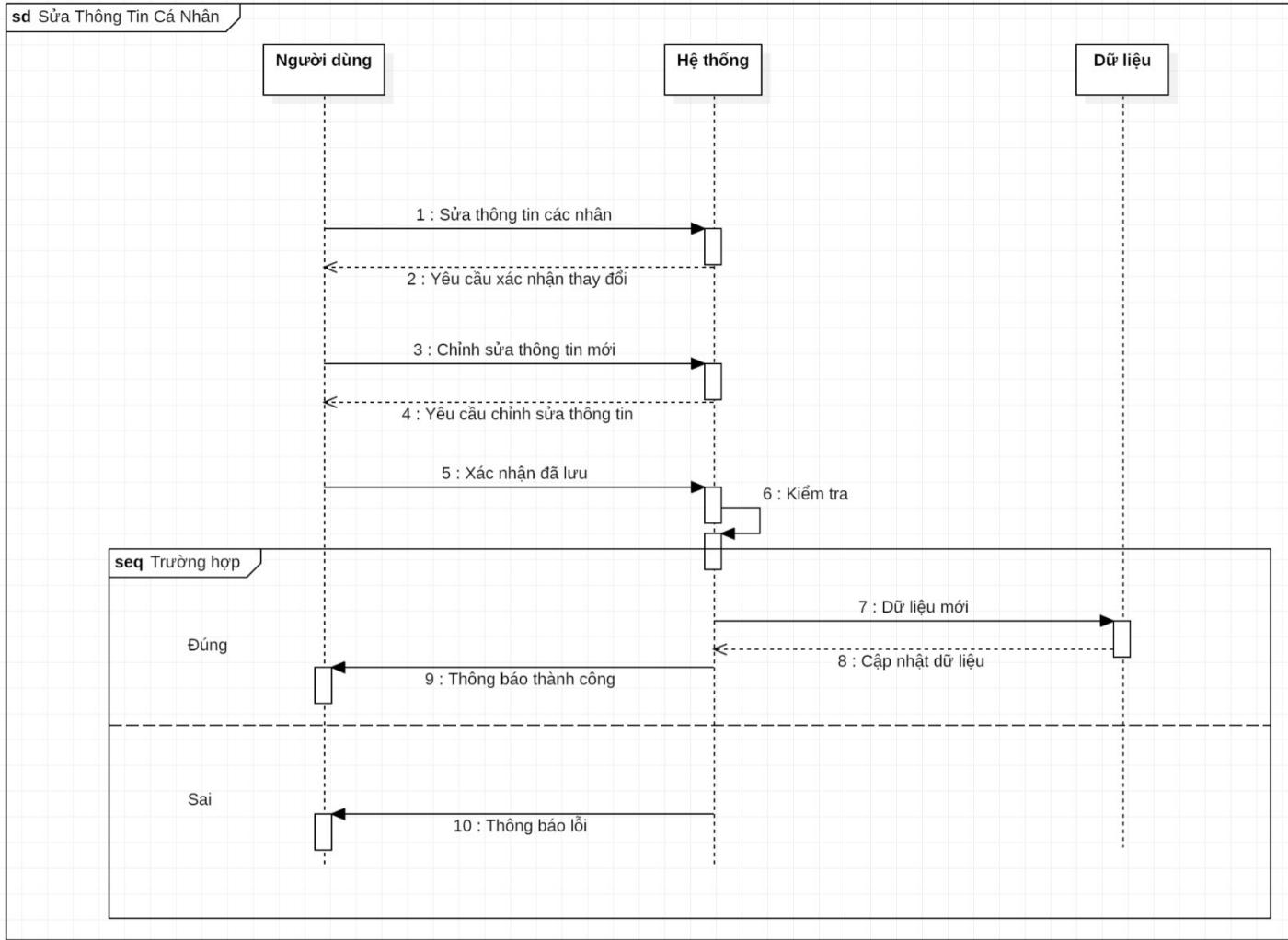
Hình 3-9 Quy trình đăng ký tài khoản người dùng

1.1.2 Quy trình đăng nhập của người dùng



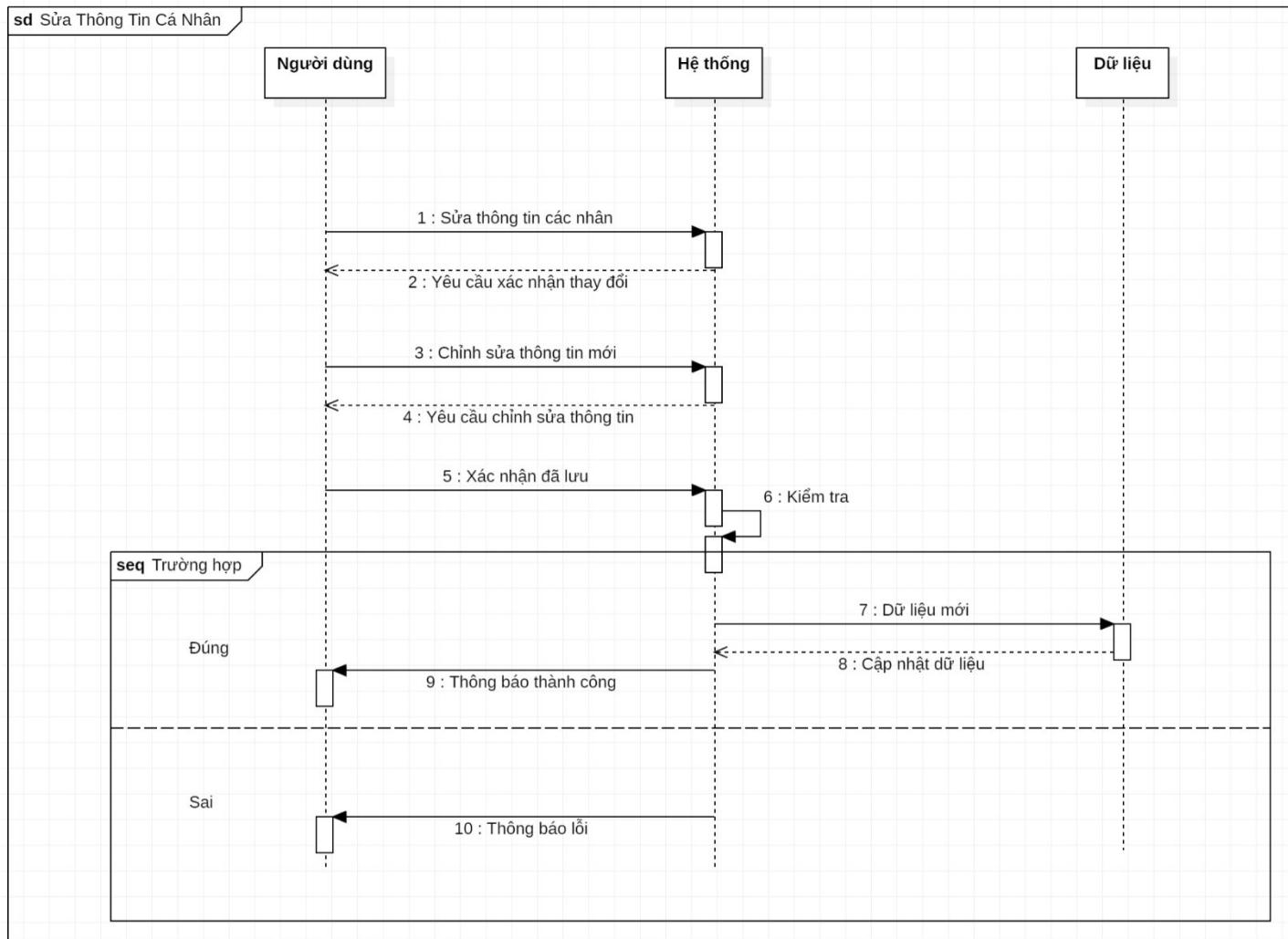
Hình 3-10 Quy trình đăng nhập của người dùng

1.1.3 Quy trình sửa thông tin cá nhân



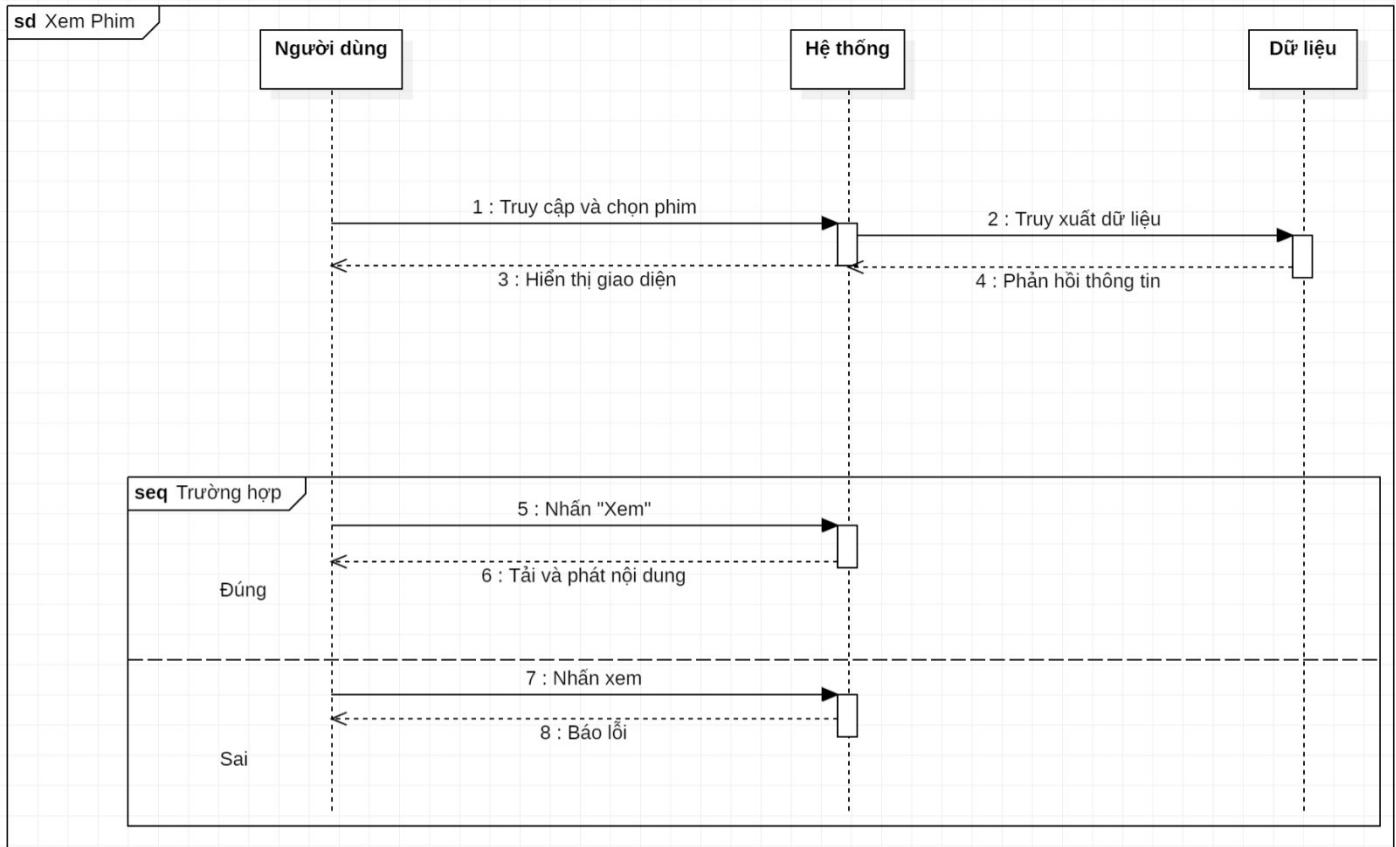
Hình 3-11 Quy trình sửa thông tin cá nhân

1.1.4 Quy trình tìm kiếm phim



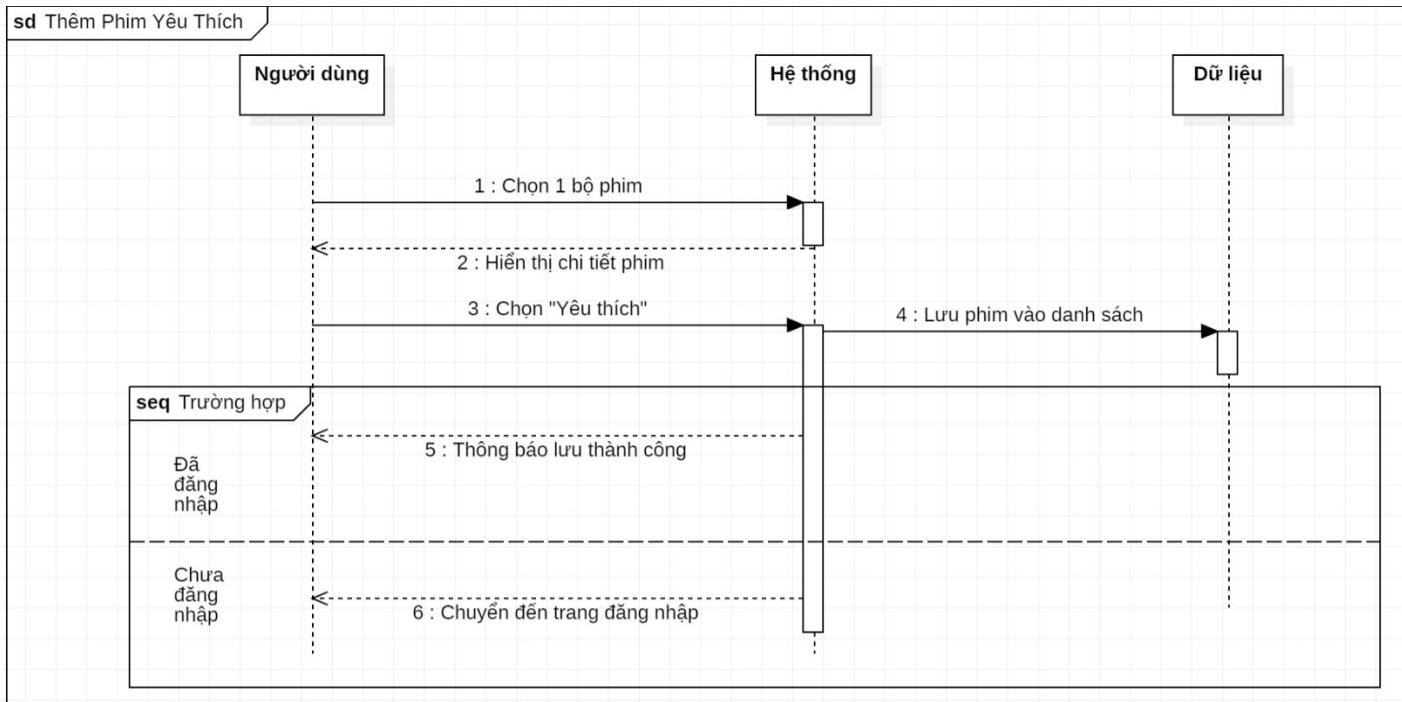
Hình 3-12 Quy trình tìm kiếm phim

1.1.5 Quy trình xem phim



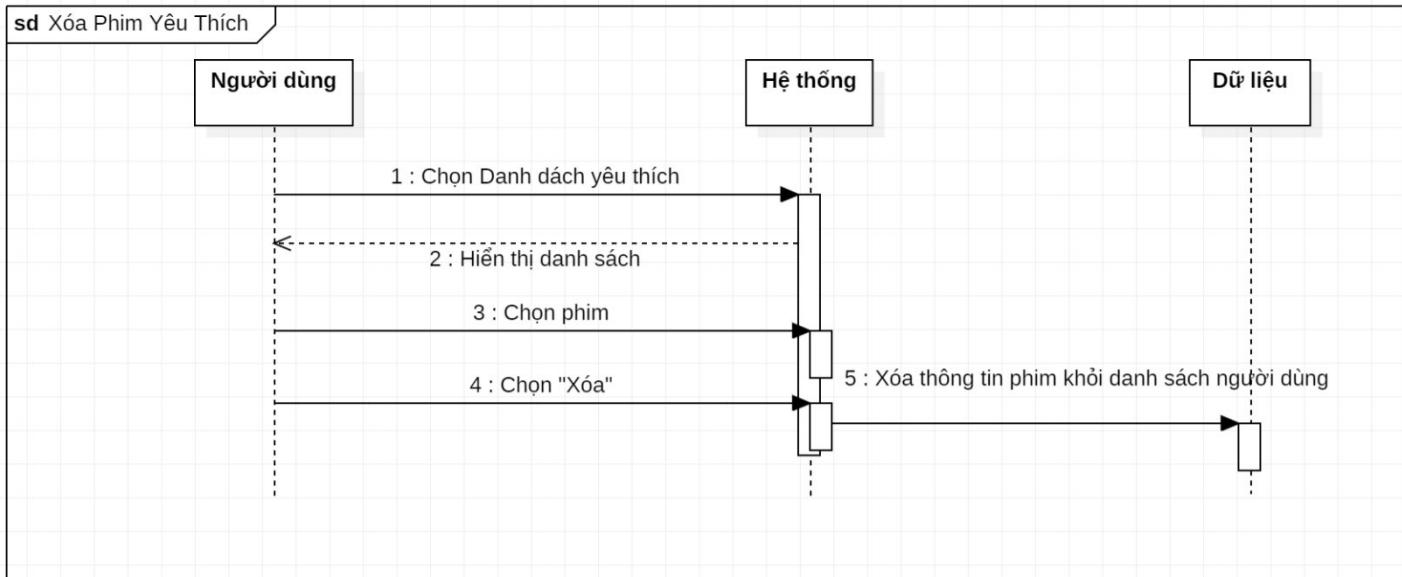
Hình 3-13 Quy trình xem phim

1.1.6 Quy trình thêm phim yêu thích



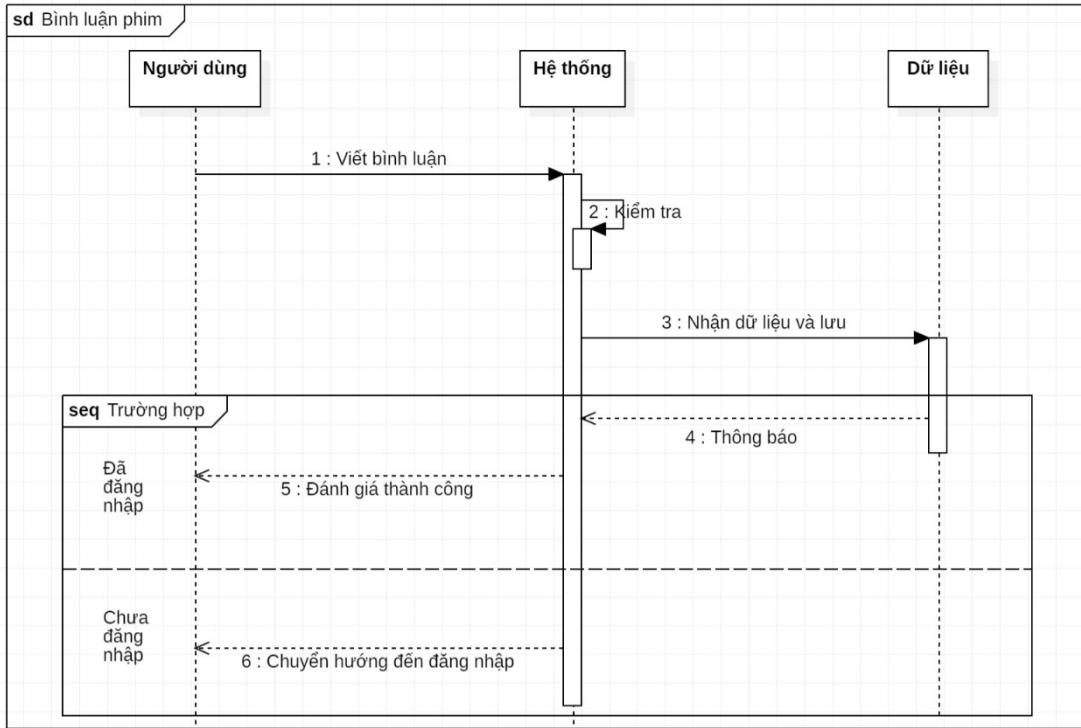
Hình 3-14 Quy trình thêm phim yêu thích

1.1.7 Quy trình xóa phim yêu thích



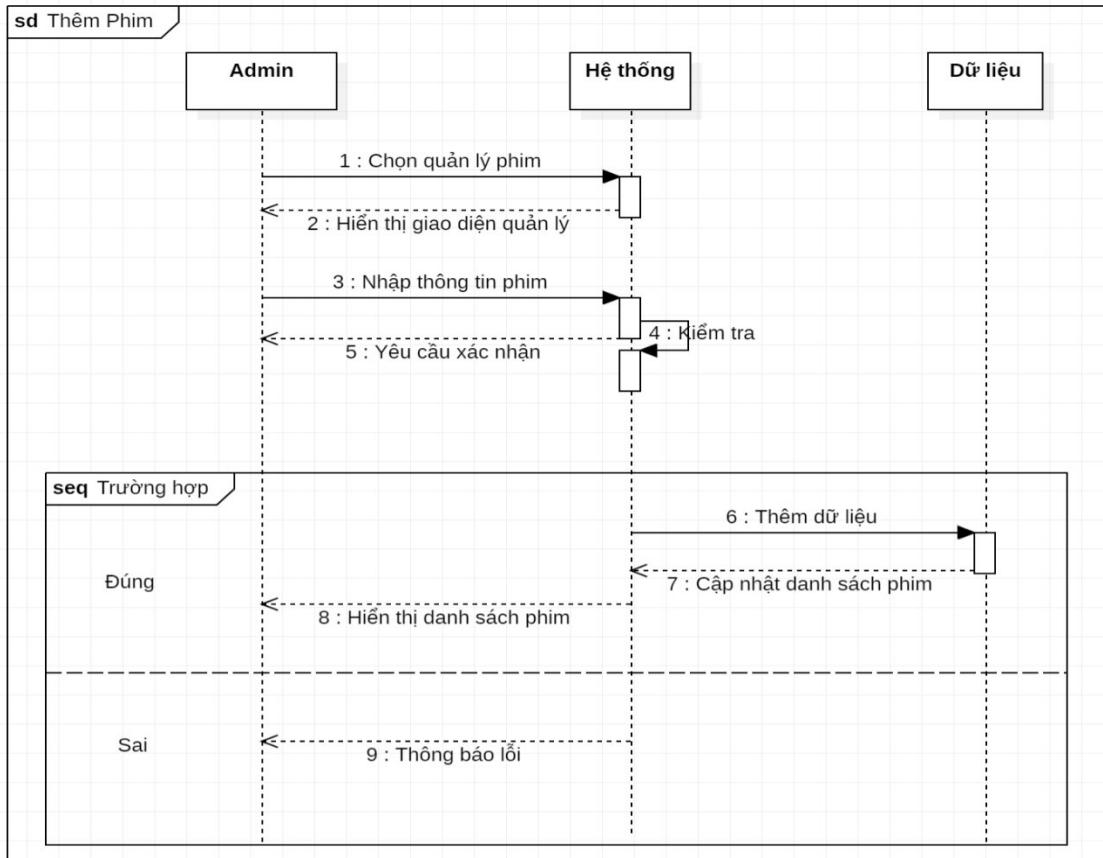
Hình 3-15 Quy trình xóa phim yêu thích

1.1.8 Quy trình bình luận phim



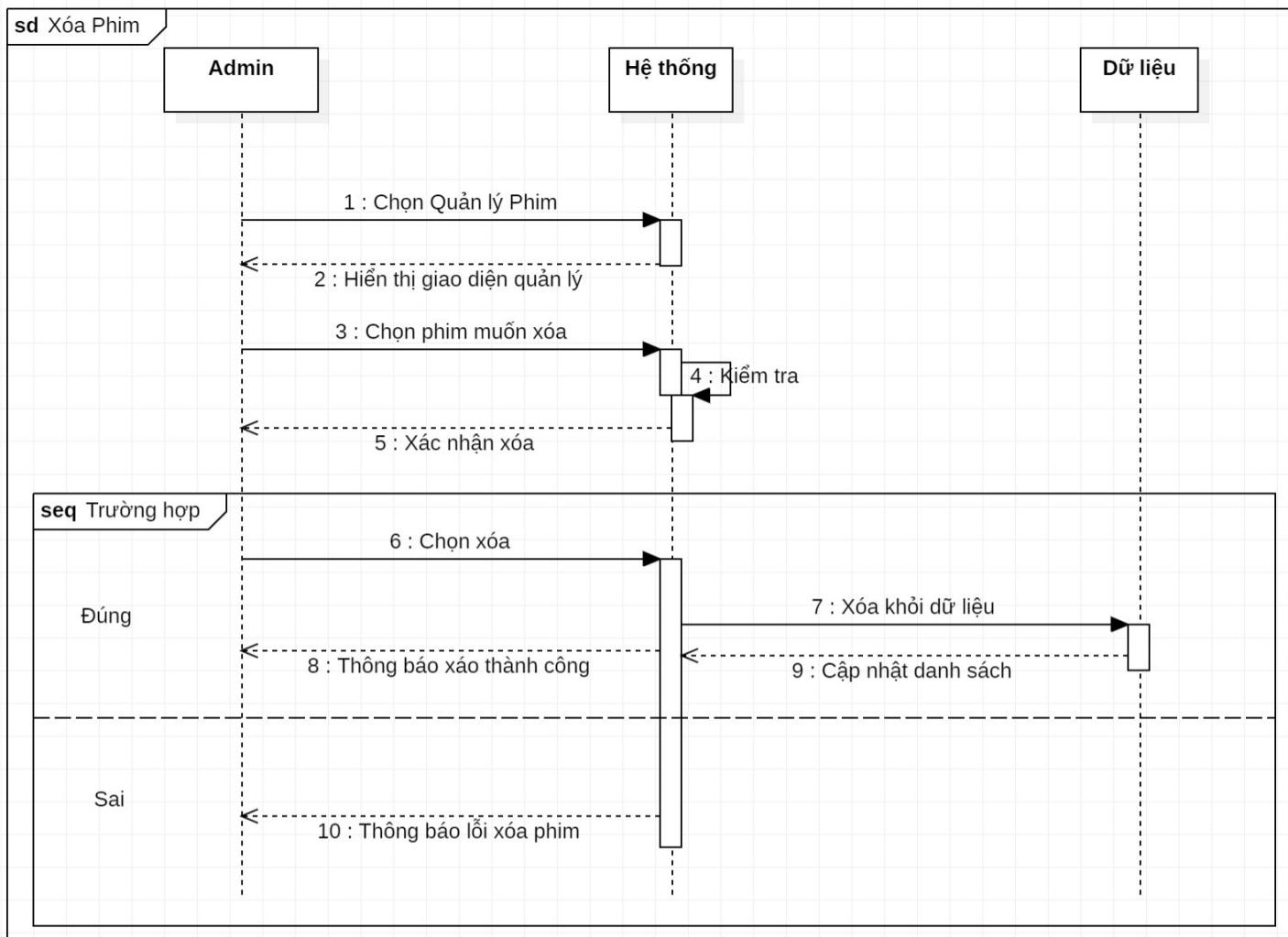
Hình 3-16 Quy trình bình luận phim

1.1.9 Quy trình thêm phim



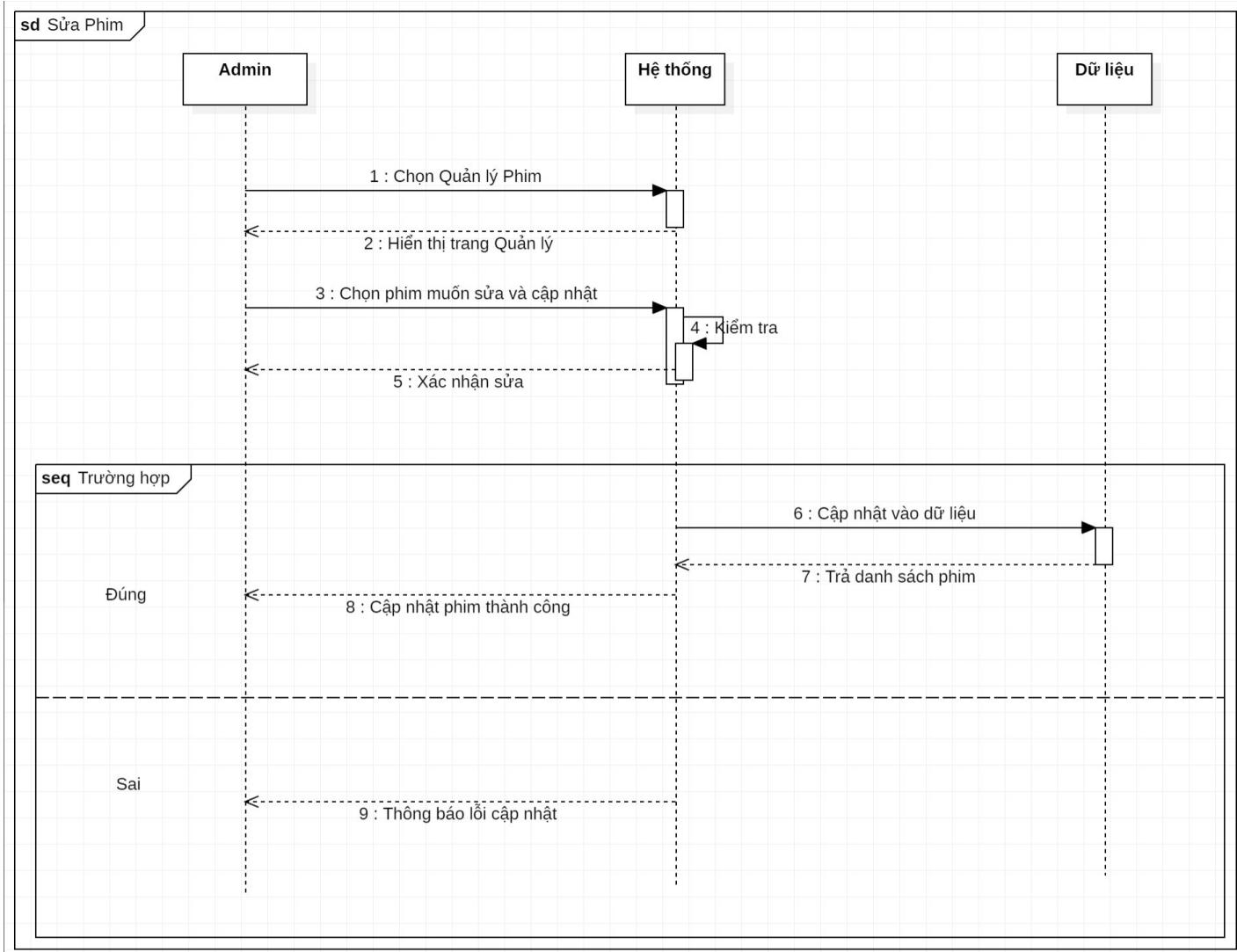
Hình 3-17 Quy trình thêm phim

1.1.10 Quy trình xóa phim



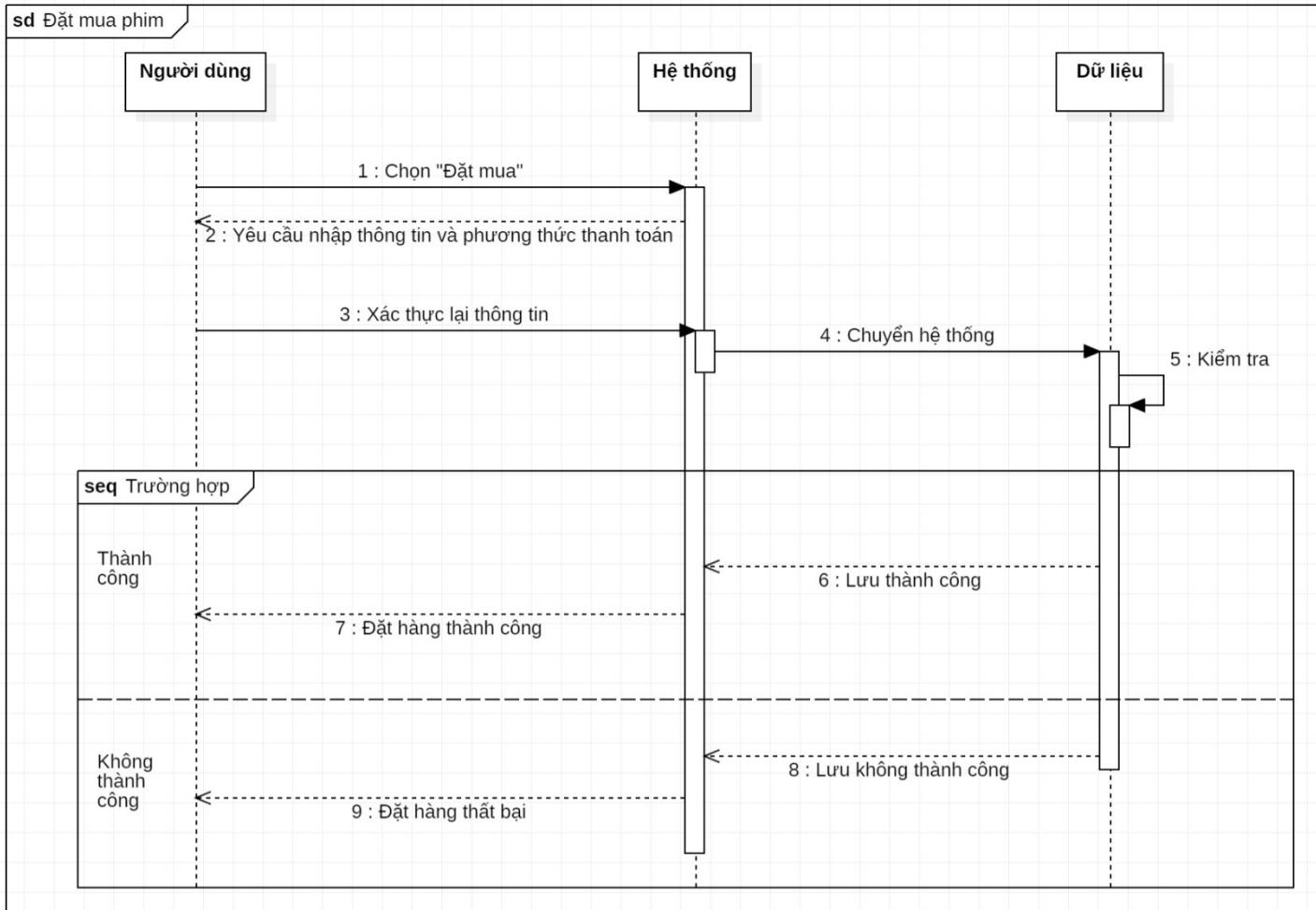
Hình 3-18 Quy trình xóa phim

1.1.11 Quy trình sửa phim



Hình 3-19 Quy trình sửa phim

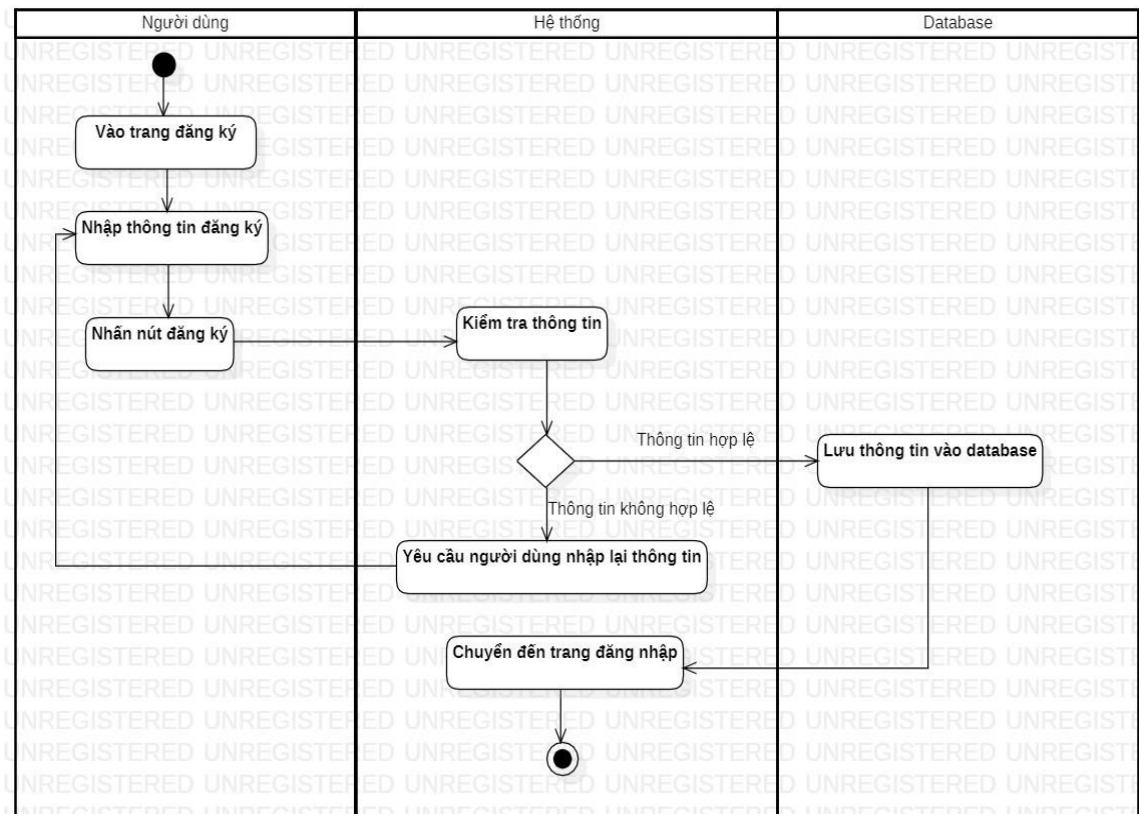
1.1.12 Quy trình đặt mua phim



Hình 3-20 Quy trình đặt mua phim

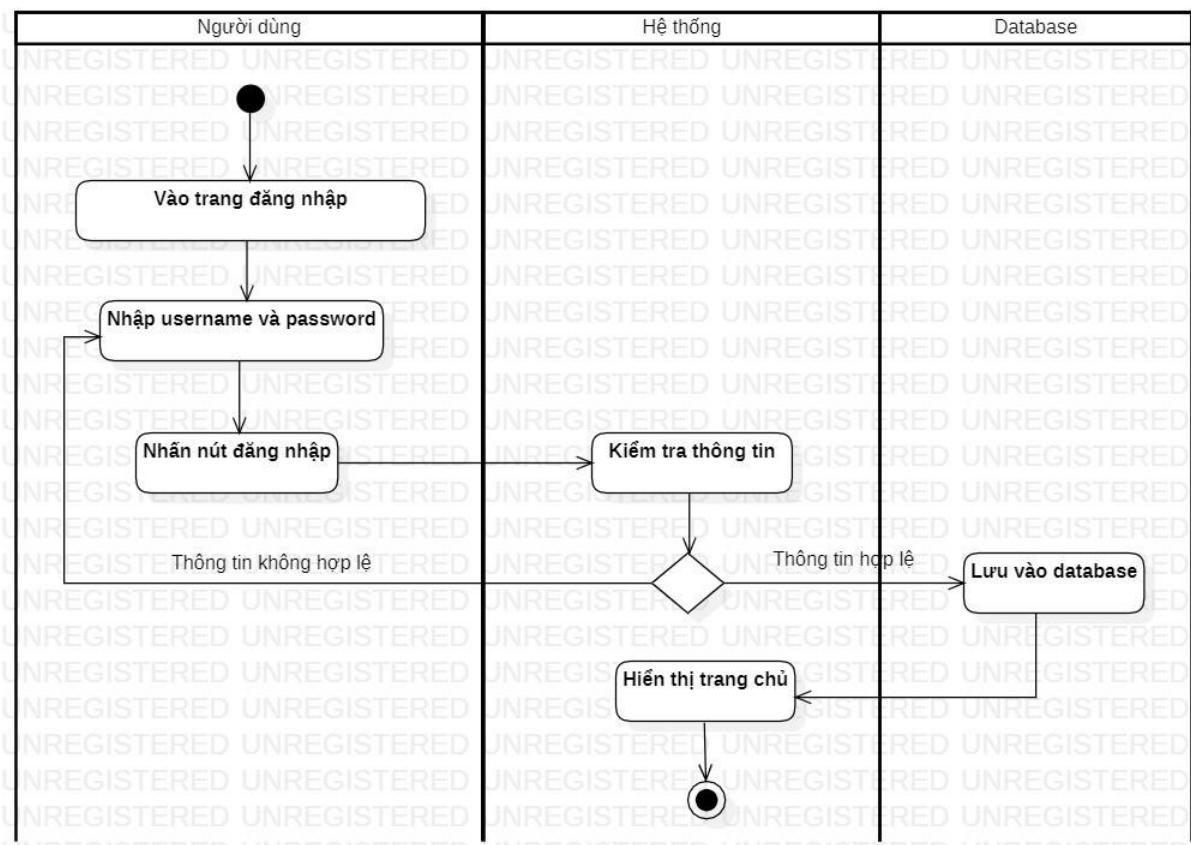
1.2 Activity Diagram

1.2.1 Quy trình Đăng ký



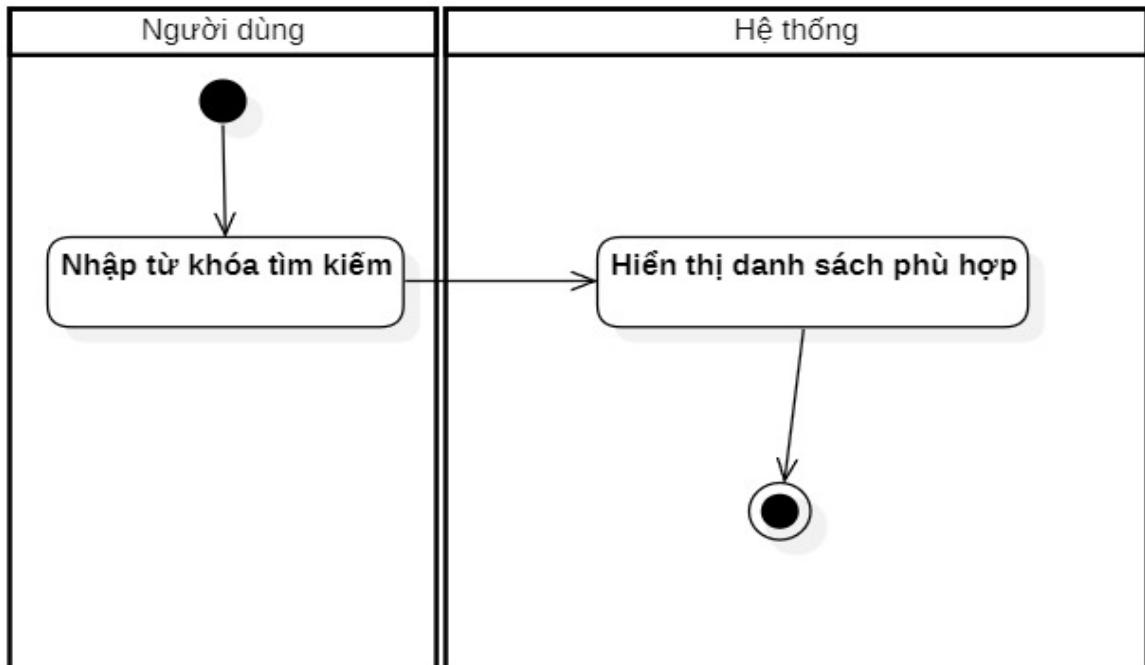
Hình 3-21 Activity Đăng ký

1.2.2 Quy trình Đăng nhập



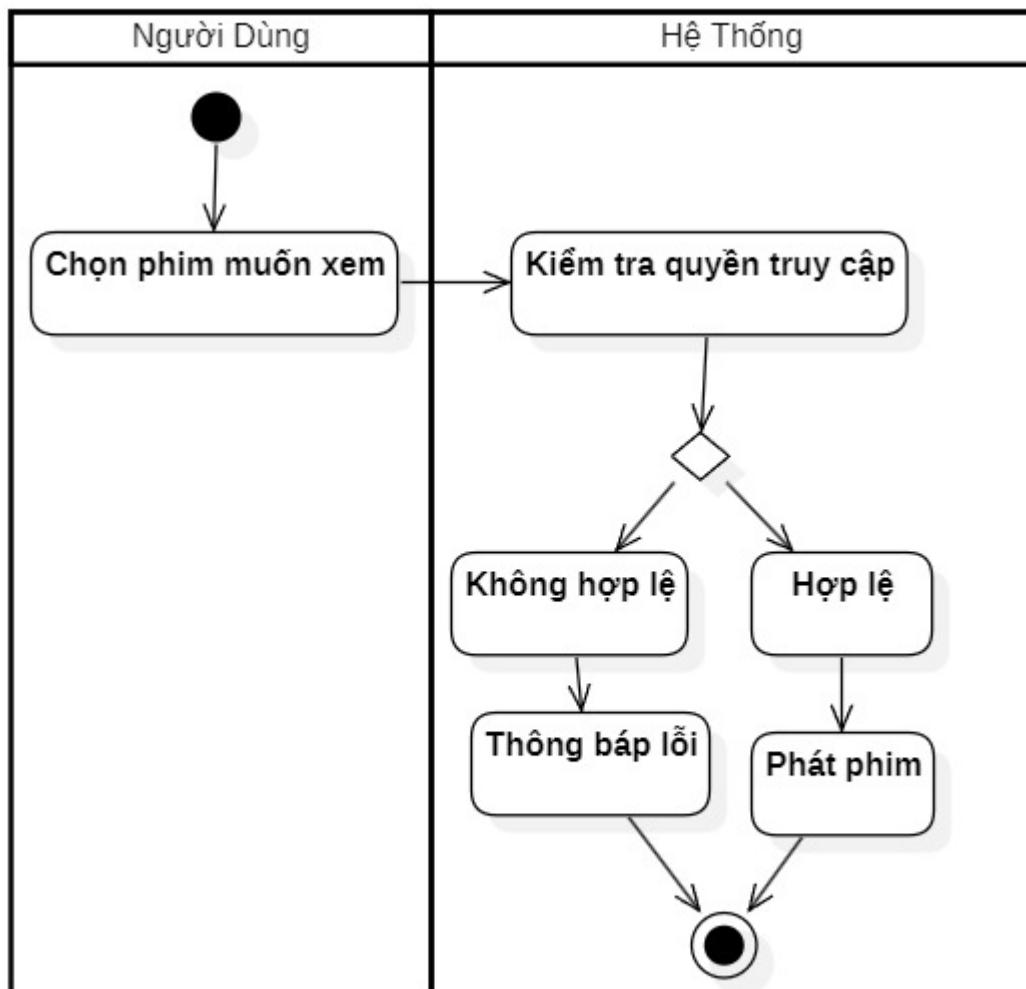
Hình 3-22 Activity Đăng nhập

1.2.3 Quy trình tìm kiếm và lọc phim



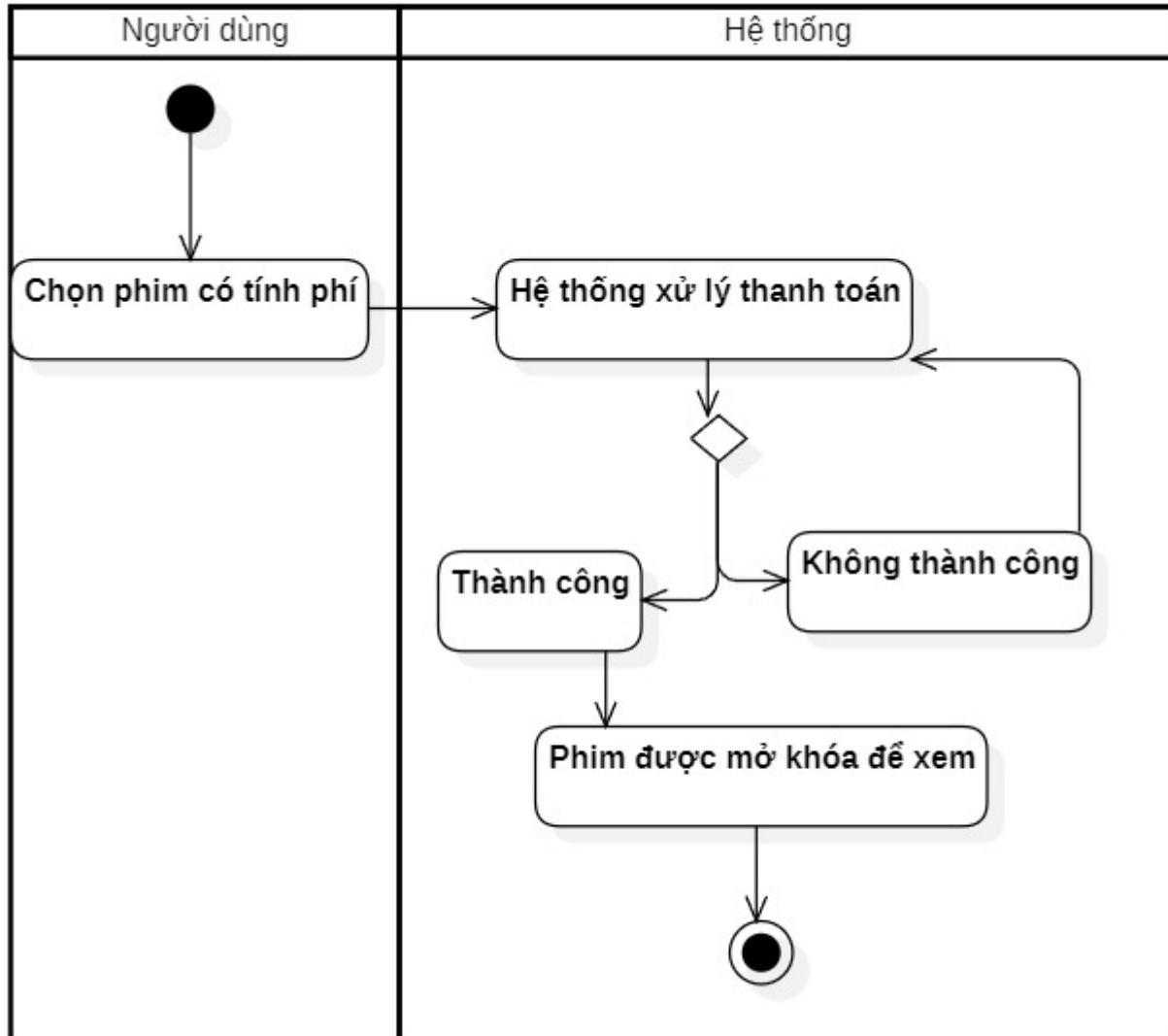
Hình 3-23 Activity tìm kiếm và lọc phim

1.2.4 Quy trình Chọn phim



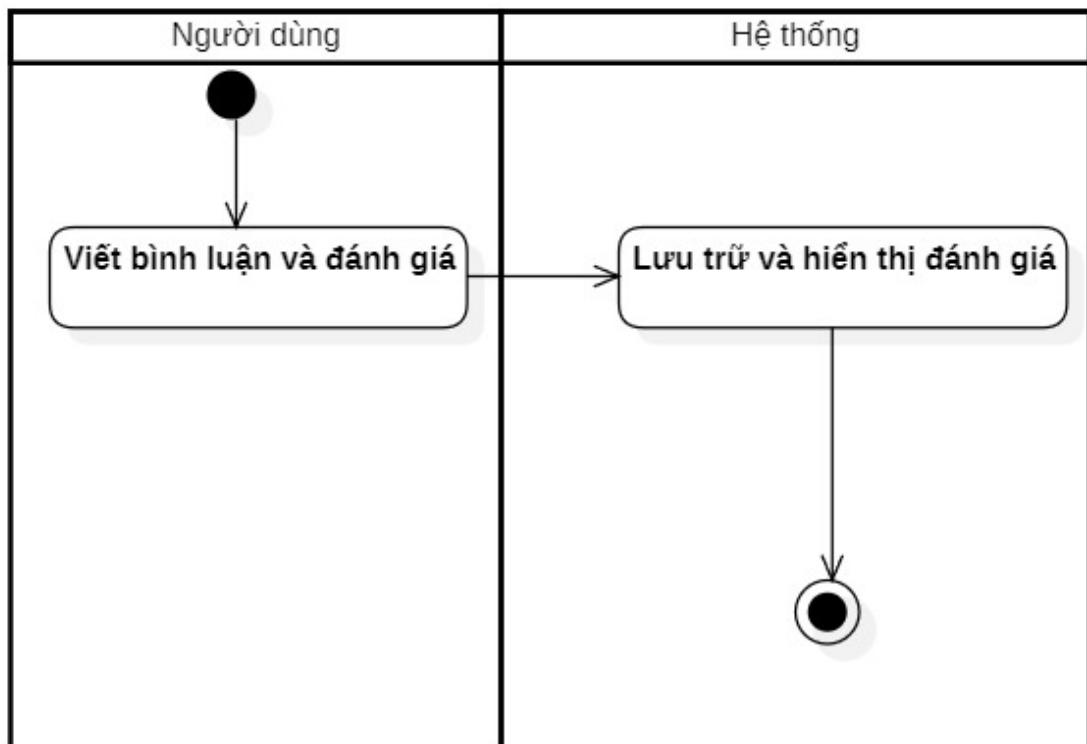
Hình 3-24 Activity cho chọn phim

1.2.5 Quy trình Thanh toán



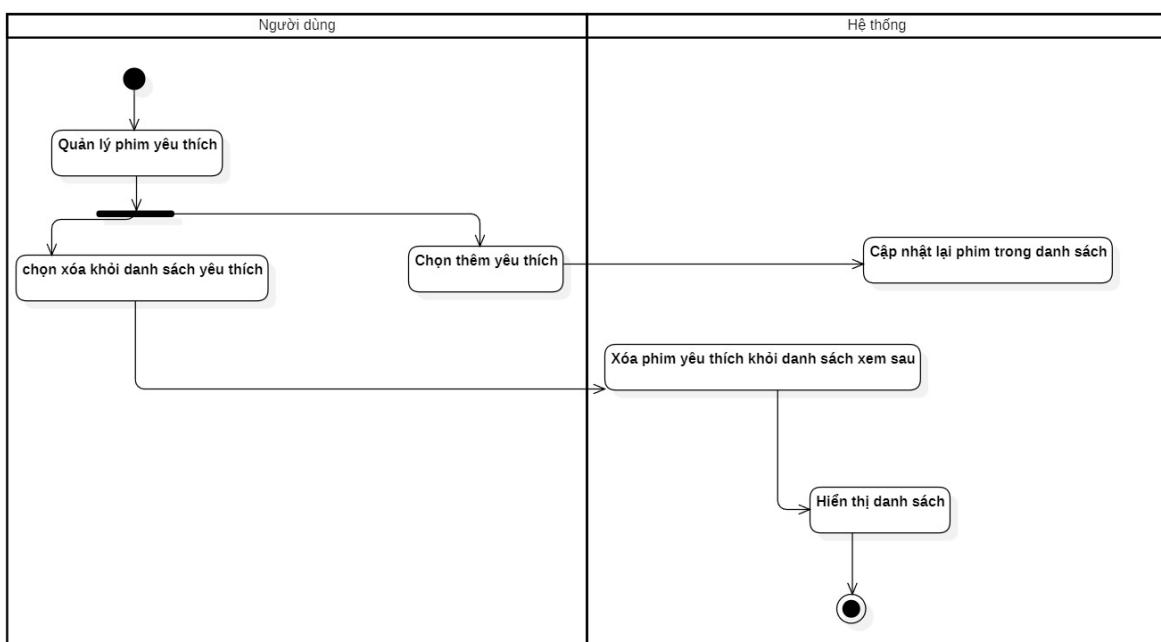
Hình 3-25 Activity Thanh toán

1.2.6 Quy trình đánh giá và bình luận



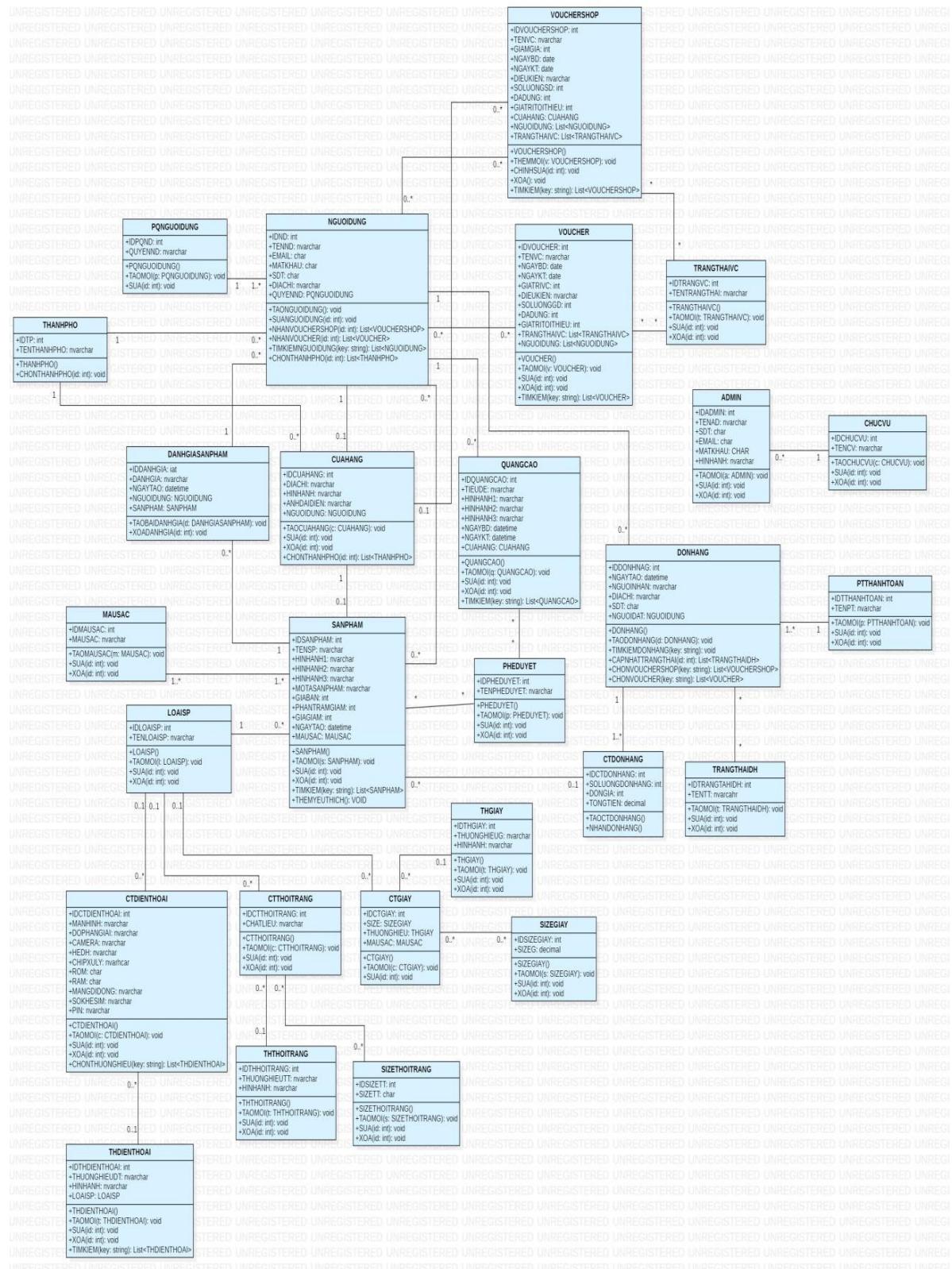
Hình 3-26 Activity đánh giá và bình luận

1.2.7 Quy trình Quản lý danh sách phim yêu thích



Hình 3-27 Activity Quản lý danh sách phim yêu thích

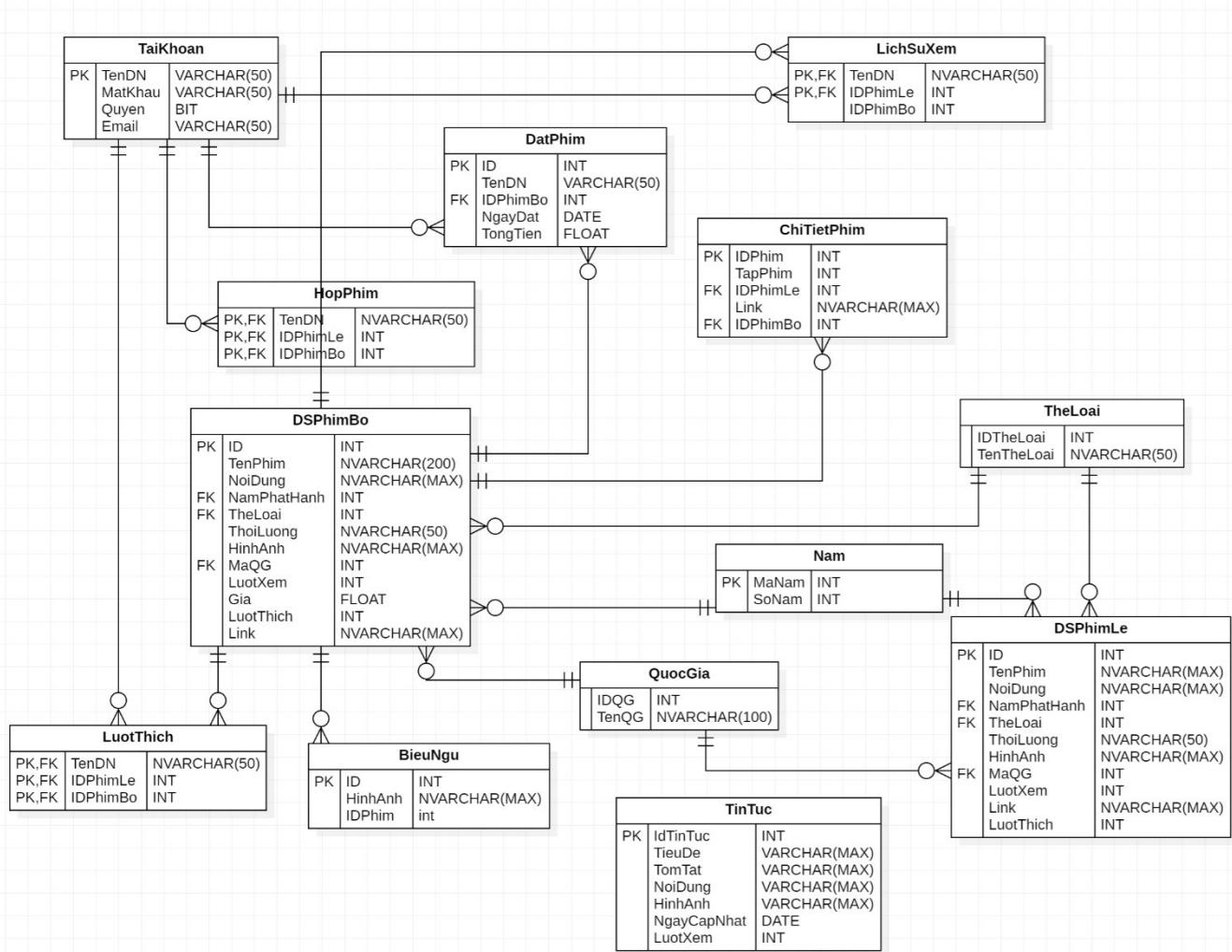
1.3 Class Diagram



Hinh 3-28 Class Diagram

4. THIẾT KẾ DỮ LIỆU

4.1. Sơ đồ logic



Hình 4-1 Sơ đồ logic

4.2. Chi tiết các bảng

4.2.1. Bảng TaiKhoan (Tài khoản người dùng):

Bảng 4-1 Bảng TaiKhoan (Tài khoản người dùng):

STT	Thuộc tính	Điễn giải	Kiểu	Ràng buộc
1	TenDN	Tên đăng nhập	Chuỗi	Khóa chính
2	MatKhau	Mật khẩu tài khoản	Chuỗi	Không được phép trùng
3	Quyen	Phân quyền đăng nhập	Bit	Không được phép rỗng
4	Email	Email liên hệ	Chuỗi	Không được phép rỗng

4.2.2. Bảng DSPhimBo (Danh sách phim bộ):

Bảng 4-2 Bảng DSPhimBo (Danh sách phim bộ):

STT	Thuộc tính	Điễn giải	Kiểu	Ràng buộc
1	ID	Tên đăng nhập	Số nguyên	Khóa chính, khóa ngoại
2	TenPhim	Mật khẩu tài khoản	Chuỗi	Không được phép rỗng, trùng
3	NoiDung	Phân quyền đăng nhập	Chuỗi	Không được phép rỗng
4	NamPhatHanh	Email liên hệ	Số nguyên	Không được phép rỗng
5	IDTheLoai	Mã thể loại phim	Số nguyên	Không được phép rỗng
6	ThoiLuong	Thời lượng trình chiếu phim	Chuỗi	Không được phép rỗng
7	Img	Hình ảnh đại diện phim	Chuỗi	
8	MaQG	Mã quốc gia sản xuất phim	Số nguyên	Không được phép rỗng
9	LuotXem	Số lượt truy cập xem phim	Số nguyên	

4.2.3. Bảng CTTapPhim (Chi tiết tập phim bộ)

Bảng 4-3 Bảng CTTapPhim (Chi tiết tập phim bộ)

STT	Thuộc tính	Điễn giải	Kiểu	Ràng buộc
1	IDPhim	Mã tập phim	Số nguyên	Khóa chính
2	TapPhim	Tên tập phim	Số nguyên	Không được phép rỗng
3	ID	Mã phim bộ	Số nguyên	Không được phép rỗng
4	Link	Link đường dẫn tập phim của phim bộ	Chuỗi	Không được phép rỗng

4.2.4. Bảng DSPhimLe (Danh sách phim lẻ)

Bảng 4-4 Bảng DSPhimLe (Danh sách phim lẻ)

STT	Thuộc tính	Điễn giải	Kiểu	Ràng buộc
1	ID	Tên đăng nhập	Số nguyên	Khóa chính
2	TenPhim	Mật khẩu tài khoản	Chuỗi	Không được phép rỗng, trùng
3	NoiDung	Phân quyền đăng nhập	Chuỗi	Không được phép rỗng
4	NamPhatHanh	Email liên hệ	Số nguyên	Không được phép rỗng
5	IDTheLoai	Mã thể loại phim	Số nguyên	Không được phép rỗng
6	ThoiLuong	Thời lượng trình chiếu phim	Chuỗi	Không được phép rỗng
7	Img	Hình ảnh đại diện phim	Chuỗi	
8	MaQG	Mã quốc gia sản xuất phim	Số nguyên	Không được phép rỗng
9	LuotXem	Số lượt truy cập	Số nguyên	

		xem phim		
10	Link	Link đường dẫn lưu trữ phim	Chuỗi	Không được phép rỗng

4.2.5. Bảng HopPhim (Hộp phim xem sau)

Bảng 4-5 Bảng HopPhim (Hộp phim xem sau)

STT	Thuộc tính	Điễn giải	Kiểu	Ràng buộc
1	TenDN	Tên đăng nhập tài khoản người dùng	Số nguyên	Khóa chính
2	IDPhim	Mã phim xem sau (Phim bộ và phim lẻ)	Số nguyên	Không được phép rỗng

4.2.6. Bảng LichSuPhim (Lịch sử xem phim)

Bảng 4-6 Bảng LichSuPhim (Lịch sử xem phim)

STT	Thuộc tính	Điễn giải	Kiểu	Ràng buộc
1	TenDN	Tên đăng nhập tài khoản người dùng	Số nguyên	Khóa chính
2	IDPhim	Mã phim đã xem (Phim bộ và lẻ)	Số nguyên	Không được phép rỗng

4.2.7. Bảng TheLoai (Thể loại)

Bảng 4-7 Bảng TheLoai (Thể loại)

STT	Thuộc tính	Điễn giải	Kiểu	Ràng buộc
1	IDTheLoai	Mã thể loại	Số nguyên	Khóa chính, khóa ngoại
2	TenTheLoai	Tên thể loại phim	Chuỗi	Không được phép rỗng

4.2.8. Bảng QuocGia (Quốc gia)

Bảng 4-8 Bảng QuocGia (Quốc gia)

STT	Thuộc tính	Điễn giải	Kiểu	Ràng buộc
1	MaQG	Mã quốc gia	Số nguyên	Khóa chính, khóa ngoại
2	TenQG	Tên quốc gia	Chuỗi	Không được phép rỗng, trùng

4.2.9. Bảng Nam (Năm phát hành phim)

Bảng 4-9 Bảng Nam (Năm phát hành phim)

STT	Thuộc tính	Điễn giải	Kiểu	Ràng buộc
1	MaNam	Mã năm	Số nguyên	Khóa chính
2	TenNam	Tên năm	Chuỗi	Không được phép rỗng, trùng

4.2.10. Bảng Banner (Banner quảng bá phim bộ)

Bảng 4-10 Bảng Banner (Banner quảng bá phim bộ)

STT	Thuộc tính	Điễn giải	Kiểu	Ràng buộc
1	ID	Mã năm	Số nguyên	Khóa chính
2	Img	Tên hình ảnh của Banner	Chuỗi	
3	IDPhim	Mã phim bộ	Số nguyên	Không được phép rỗng

4.2.11. Bảng tintucphim (Tin tức về phim điện ảnh)

Bảng 4-11 Bảng tintucphim (Tin tức về phim điện ảnh)

STT	Thuộc tính	Điễn giải	Kiểu	Ràng buộc
1	idtintuc	Mã bài tin tức	Số nguyên	Khóa chính
2	tieude	Tiêu đề tin tức	Chuỗi	Không được phép rỗng
3	tomtat	Tóm tắt nội dung tin tức	Chuỗi	Không được phép rỗng
4	noidung	Nội dung chi tiết của tin tức	Chuỗi	Không được phép rỗng
5	hinhanh	Hình ảnh mô tả đại diện	Chuỗi	
6	ngaycapnhat	Ngày cập nhật đăng bài	Ngày	Không được phép rỗng
7	luotxem	Số lượt	Số	

		truy cập xem tin tức	nguyên	
--	--	-------------------------	--------	--

4.2.12. Bảng gioithieu (Giới thiệu trang website)

Bảng 4-12 Bảng gioithieu (Giới thiệu trang website)

STT	Thuộc tính	Điễn giải	Kiểu	Ràng buộc
1	idgioithieu	Mã bài giới thiệu	Số nguyên	Khóa chính
2	noidung	Nội dung bài giới thiệu	Chuỗi	
3	sdtlienhe	Số điện thoại liên hệ của người đại diện Website	Chuỗi	

4.3. Các câu lệnh trigger và stored procedures:

4.3.1. Trigger:

```
***** Trigger để cập nhật lượt xem khi người dùng xem phim *****/
CREATE TRIGGER trg_UpdateViewCount
ON LichSu
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE DSPhimBo
    SET LuotXem = LuotXem + 1
    WHERE ID IN (SELECT IDPhim FROM inserted);

    UPDATE DSPhimLe
    SET LuotXem = LuotXem + 1
    WHERE ID IN (SELECT IDPhim FROM inserted);
END;
GO
```

Hình 4-2 Trigger

4.3.2. Stored Procedures:

```
***** Trigger để cập nhật lượt xem khi người dùng xem phim *****/
CREATE TRIGGER trg_UpdateViewCount
ON LichSu
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE DSPhimBo
    SET LuotXem = LuotXem + 1
    WHERE ID IN (SELECT IDPhim FROM inserted);

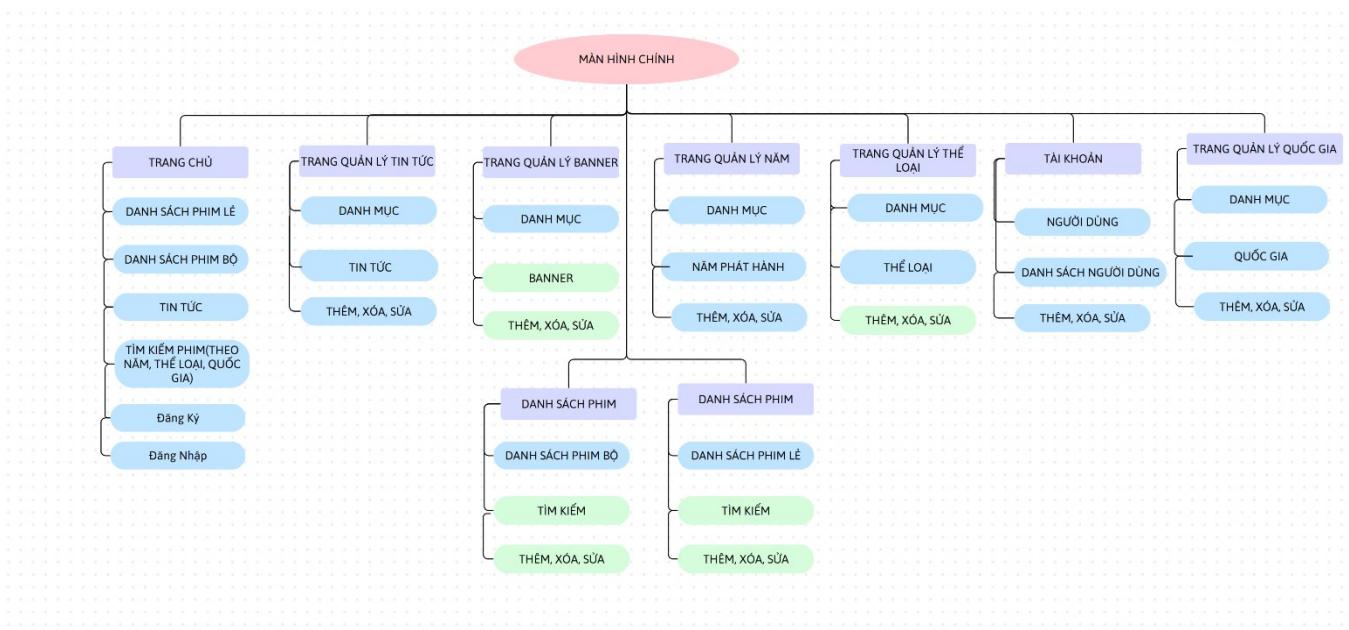
    UPDATE DSPhimLe
    SET LuotXem = LuotXem + 1
    WHERE ID IN (SELECT IDPhim FROM inserted);
END;
GO
```

Hình 4-3 Stored Procedures:

5. THIẾT KẾ GIAO DIỆN

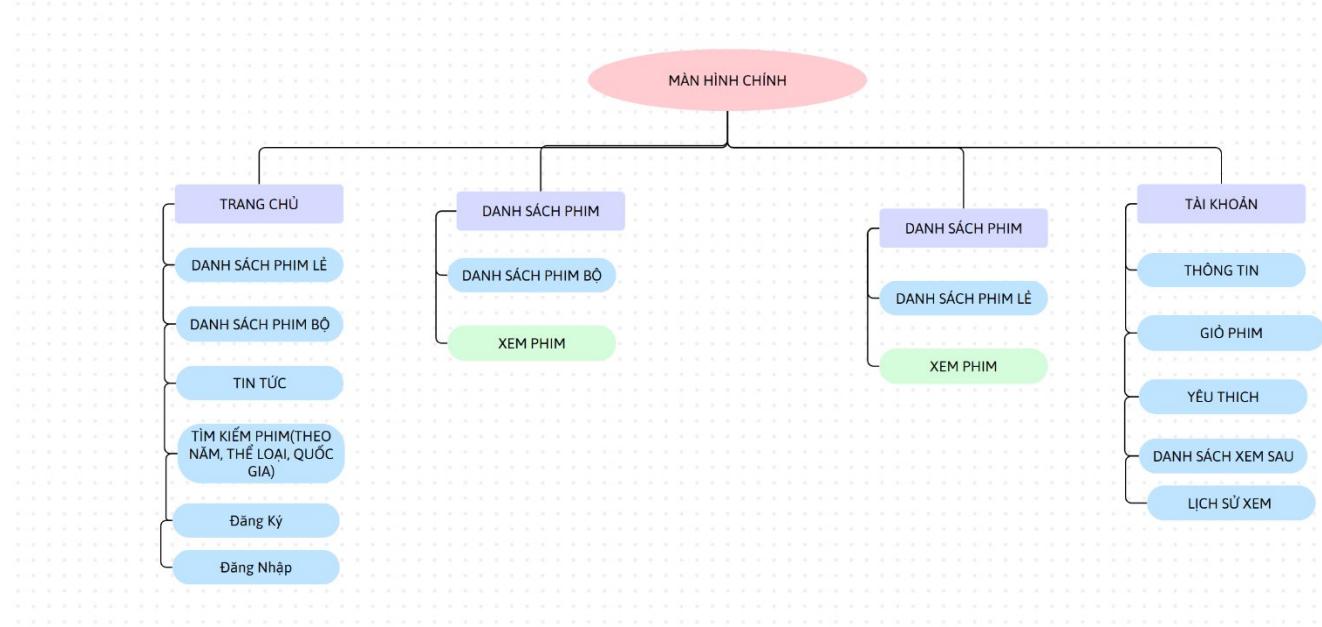
5.1.1. Sơ đồ giao diện tổng quát

5.1.1. Sơ đồ giao diện tổng quát Admin



Hình 5-1 Sơ đồ giao diện tổng quát Admin

5.1.2. Sơ đồ giao diện tổng quát Người xem



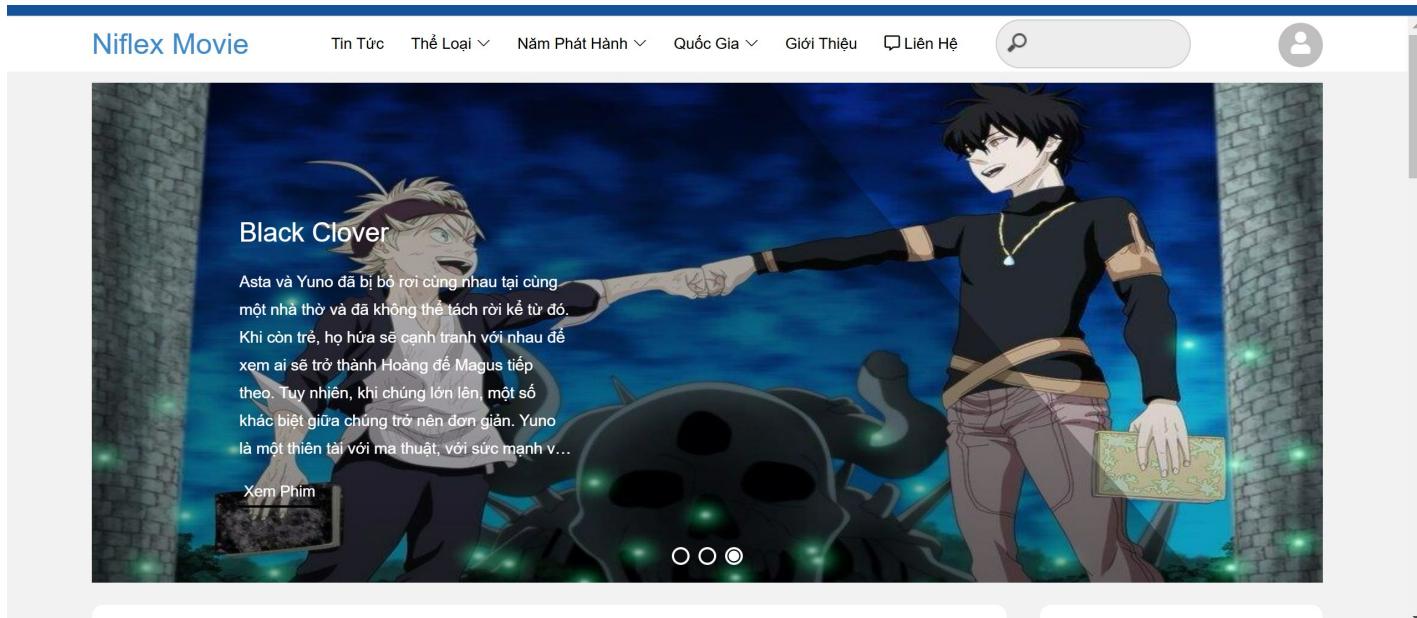
Hình 5-2 Sơ đồ giao diện tổng quát Người xem

5.1. Giao diện chi tiết

5.2.1 Người dùng

Trang chủ

Sau khi người dùng truy cập vào trang website này thì giao diện đầu tiên trang chủ sẽ hiển thị ra, sẽ có phần hiển thị danh sách các phim đang có, các tính năng lọc tìm kiếm và các bộ cục của website xem phim.



Hình 5-3 Giao diện trang chủ website xem phim của người dùng

Đăng nhập

Khi người dùng muốn sử dụng các chức năng như lưu lại phim xem sau hoặc xem lại lịch sử xem phim, đặt mua phim thì bắt buộc người dùng phải thực hiện chức năng đăng nhập vào hệ thống này. Có hai cách đăng nhập vô trang web của phim cách thứ nhất là người dùng sử dụng tài khoản mật khẩu đã đăng ký từ trước để đăng nhập, cách thứ hai là người dùng có thể sử dụng chức năng đăng nhập bằng tài khoản facebook

Đăng Nhập

Tên đăng nhập:

Mật khẩu:

Nhớ mật khẩu

ĐĂNG NHẬP

CHƯA CÓ TÀI KHOẢN? ĐĂNG KÝ

[Quên mật khẩu ?](#)

Hình 5-4 Giao diện đăng nhập tài khoản vào hệ thống

Đăng ký

Khi người dùng chưa có thông tin tài khoản để truy cập vào hệ thống website hoặc không có tài khoản để sử dụng các tính năng cao hơn thì cần phải thực hiện chức năng đăng ký thông tin tài khoản trước khi đăng nhập.

Đăng ký

Tên đăng nhập

Mật khẩu

Xác nhận mật khẩu

Email

ĐĂNG KÝ

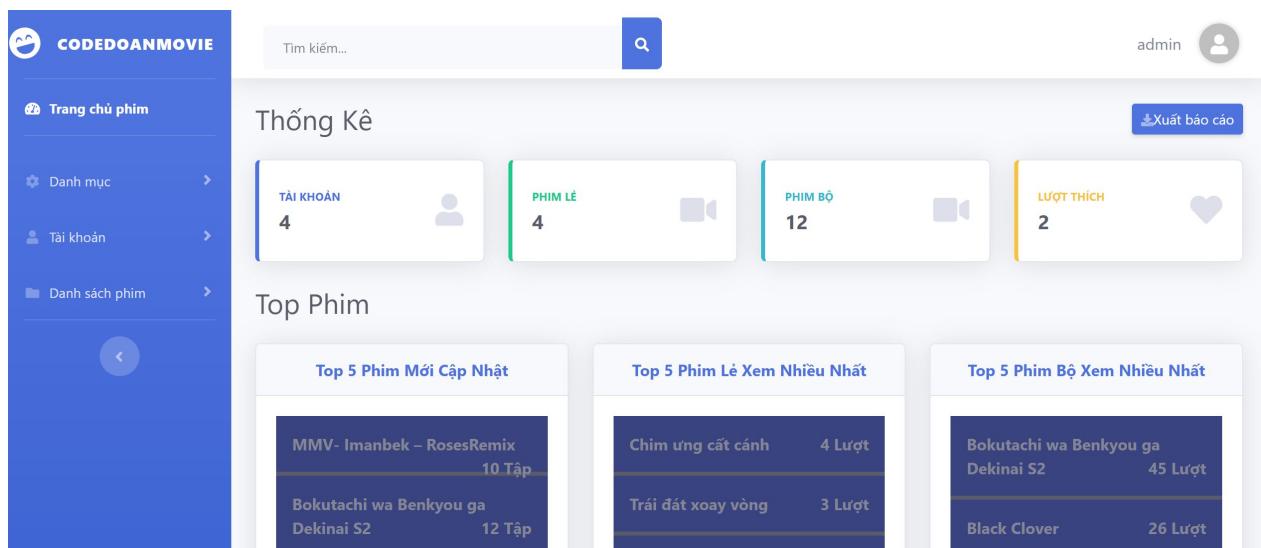
[Quay về đăng nhập](#)

Hình 5-5 Giao diện đăng ký tài khoản vào hệ thống

5.2.2 Quản trị viên website

Trang chủ quản trị

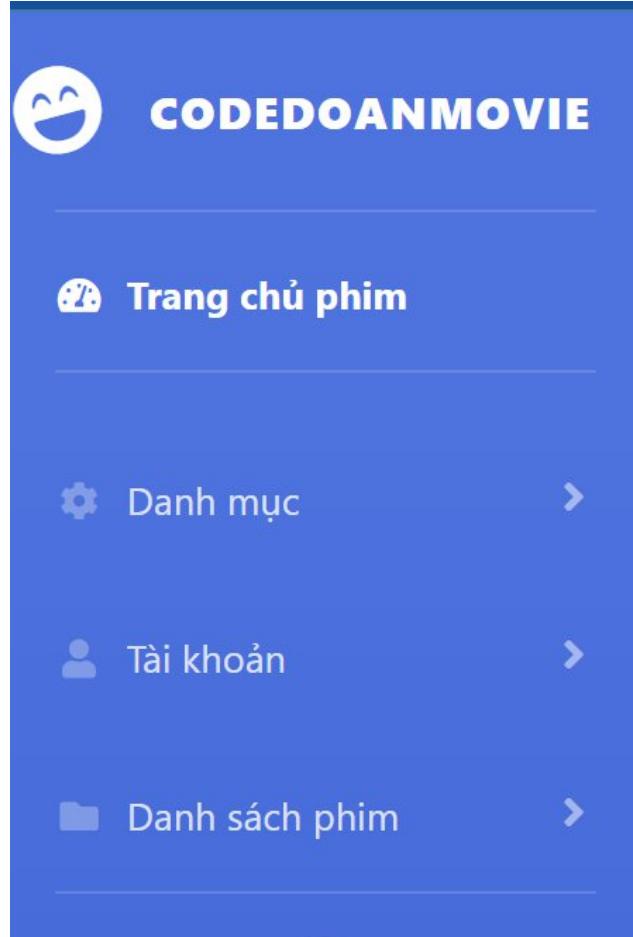
Trước khi sử dụng các tính năng của người quản trị website thì cần phải có tài khoản đăng nhập với quyền truy cập là Admin hoặc Administrator quản trị. Sau khi đăng nhập tài khoản có quyền quản trị vào thì hệ thống sẽ chuyển hướng trang web sang giao diện dành cho người quản trị hệ thống website.



Hình 5-6 Giao diện trang chủ của người quản trị (Admin và Administrator)

Quản lý

Các phần danh mục quản lý của người quản trị đều hiển thị ở danh sách menu list danh sách bên trái màn hình, người quản trị cần xem, sửa và xóa cập nhật thông tin thì chỉ cần truy cập vào một trong các danh mục quản lý ở phần menu này. Để dàng tìm kiếm và quản lý với menu dạng danh sách.

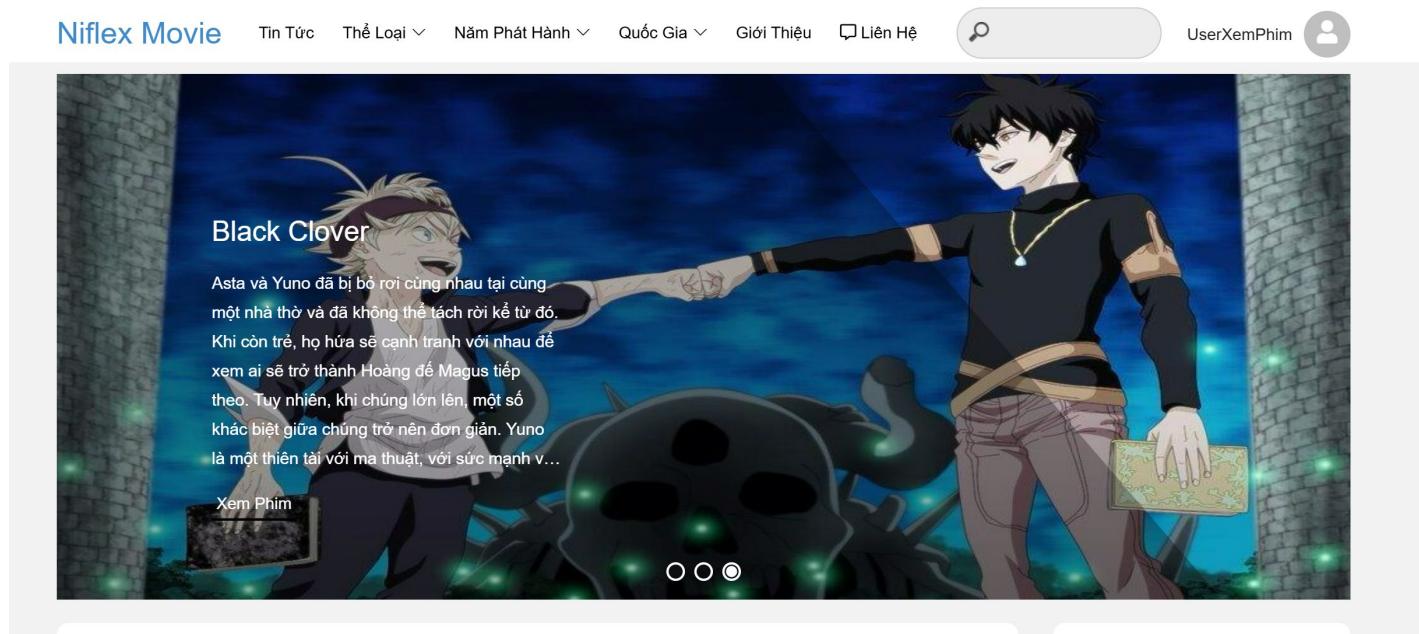


Hình 5-7 Giao diện menu quản lý danh mục của người quản trị

Người dùng

Đăng nhập, Quên mật khẩu

Bước đầu tiên để vào một hệ thống trước tiên ta cần thực hiện thao tác đăng nhập để tiến hành bắt đầu. Nếu các thông tin lưu trữ bao gồm: (tên tài khoản và mật khẩu) được nhập đúng ta sẽ giao diện sẽ hiện ra những chức năng chính của web phim Còn nếu người dùng đăng nhập thất bại ta sẽ dựa vào những sai sót để kiểm tra sau đó hiện thông báo lỗi cho người dùng. Ta có lược đồ như sau:



Hình 5-8 Giao diện trang chủ

Sau khi hiển thị ra trang chủ người dùng sẽ bấm vào biểu tượng icon hình để vào giao diện đăng nhập và nhập thông tin vào form đăng nhập. Hệ thống sẽ tiến hành kiểm tra thông tin coi người dùng nhập đúng chưa. Nếu người dùng chỉ nhập mật khẩu sau đó bấm đăng nhập mà chưa điền tên tài khoản sẽ thông báo cho người dùng thông báo lỗi như sau. Tương tự nếu người dùng chỉ nhập tài khoản mà không nhập mật khẩu ta cũng thông báo ra màn hình.

Đăng Nhập

Tên đăng nhập:

Mật khẩu:

Nhớ mật khẩu

ĐĂNG NHẬP

CHƯA CÓ TÀI KHOẢN? ĐĂNG KÝ

[Quên mật khẩu ?](#)

Đăng Nhập

Tên đăng nhập:

Mật khẩu: ! Please fill out this field.

Nhớ mật khẩu

ĐĂNG NHẬP

CHƯA CÓ TÀI KHOẢN? ĐĂNG KÝ

[Quên mật khẩu ?](#)

Đăng Nhập

Tên đăng nhập:

Mật khẩu:

Nhớ mật khẩu ! Please fill out this field.

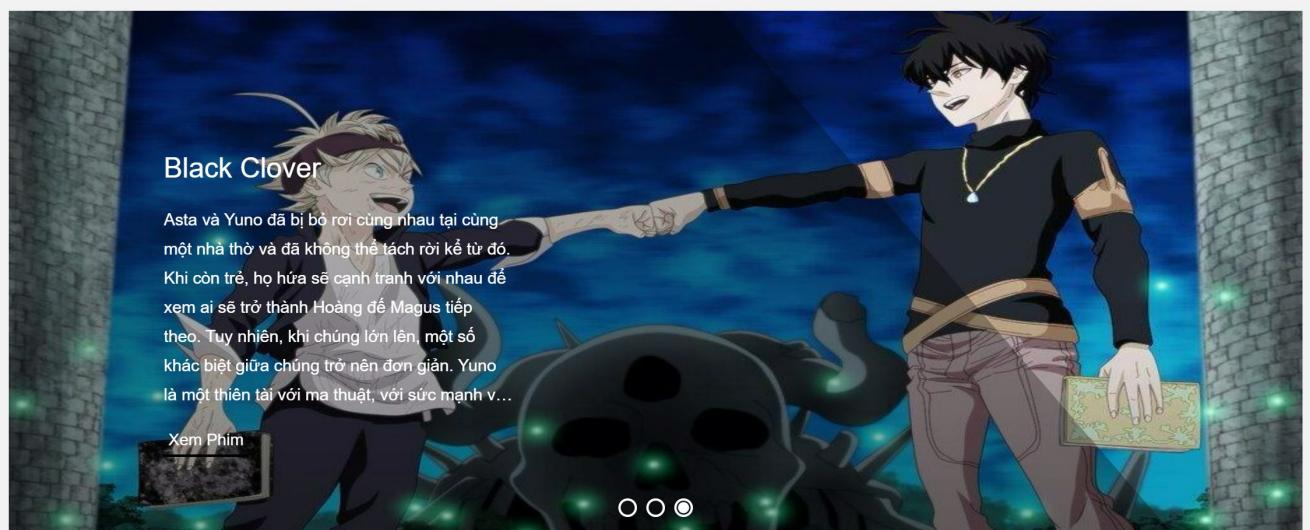
ĐĂNG NHẬP

CHƯA CÓ TÀI KHOẢN? ĐĂNG KÝ

[Quên mật khẩu ?](#)

Hình 5-9 Giao diện kiểm tra nhập liệu

Sau khi người dùng nhập đủ tên đăng nhập và tài khoản đã được đăng ký trước đó hệ thống sẽ tiến hành kiểm tra nếu đúng sẽ chuyển tới trang giao diện chính của web phim.



Hình 5-10 Giao diện trang chủ

Trong trường hợp người dùng nhập sai mật khẩu hoặc mật khẩu không đúng so với trước đó đã đăng ký trên hệ thống thì sẽ hiển thị thông báo lỗi sai mật khẩu hoặc tên đăng nhập và yêu cầu người dùng đăng nhập lại đúng thông tin để được truy cập vào hệ thống website.

Đăng Nhập

Tên đăng nhập hoặc mật khẩu không đúng!

Tên đăng nhập:

thuyduong28

Mật khẩu:

...

Nhớ mật khẩu

ĐĂNG NHẬP

CHƯA CÓ TÀI KHOẢN? ĐĂNG KÝ

[Quên mật khẩu ?](#)

Hình 5-11 Giao diện đăng nhập sai

Người dùng có thể sử dụng checkbox nhớ mật khẩu để cho lần đăng nhập tiếp theo không còn thực hiện lại việc nhập thông tin. Nếu hệ thống kiểm tra thông tin người dùng nhập không chính xác sẽ xuất ra thông báo lỗi cho người dùng ‘tên đăng nhập hoặc mật khẩu không đúng’.

Đăng ký

Nếu người dùng muốn đăng ký tài khoản thì sau khi click vào ‘Đăng ký tài khoản’ người dùng sẽ được chuyển đến trang chủ của trang đăng ký để điền vào form thông tin.

Đăng ký

Tên đăng nhập

Mật khẩu

Xác nhận mật khẩu

Email

ĐĂNG KÝ

[Quay về đăng nhập](#)

Hình 5-12 Giao diện đăng ký

Sau khi người dùng nhập những thông tin cần thiết hệ thống sẽ tiến hành truy xuất xuống cơ sở dữ liệu để kiểm tra người dùng này đã được đăng ký hay chưa. Nếu thông tin người dùng trùng với tên người dùng đã có sẵn thì sau khi nhấp vào button ‘Đăng ký’ sẽ thông báo cho người dùng ‘Đã có tài khoản này’. Ngoài ra nếu email khi tạo trùng với email của người dùng trước cũng sẽ thông báo lỗi ra cho người dùng thông tin email bị trùng’.

Đăng ký

Tên đăng nhập

Đã có tài khoản này

Mật khẩu

Xác nhận mật khẩu

Email

thông tin email bị trùng

ĐĂNG KÝ

[Quay về đăng nhập](#)

Hình 5-13 Giao diện đăng ký tài khoản trùng

Còn nếu thông tin người dùng không trùng khớp hệ thống sẽ lưu thông tin của người dùng và mã hóa mật khẩu đăng ký xuống database và chuyển người dùng đến trang chủ đăng nhập để tiến hành thao tác.

Đăng ký

Tên đăng nhập

Đã có tài khoản này

Mật khẩu

Xác nhận mật khẩu

Email

thông tin email bị trùng

ĐĂNG KÝ

[Quay về đăng nhập](#)

5	UserXemPhim	601f1889667efaebb33b8c12572835da3f027f78	0	UserXemPhim@gmail.com
6	UserXemPhim1	601f1889667efaebb33b8c12572835da3f027f78	0	UserXemPhim1@gmail.com

Tìm kiếm theo tên phim

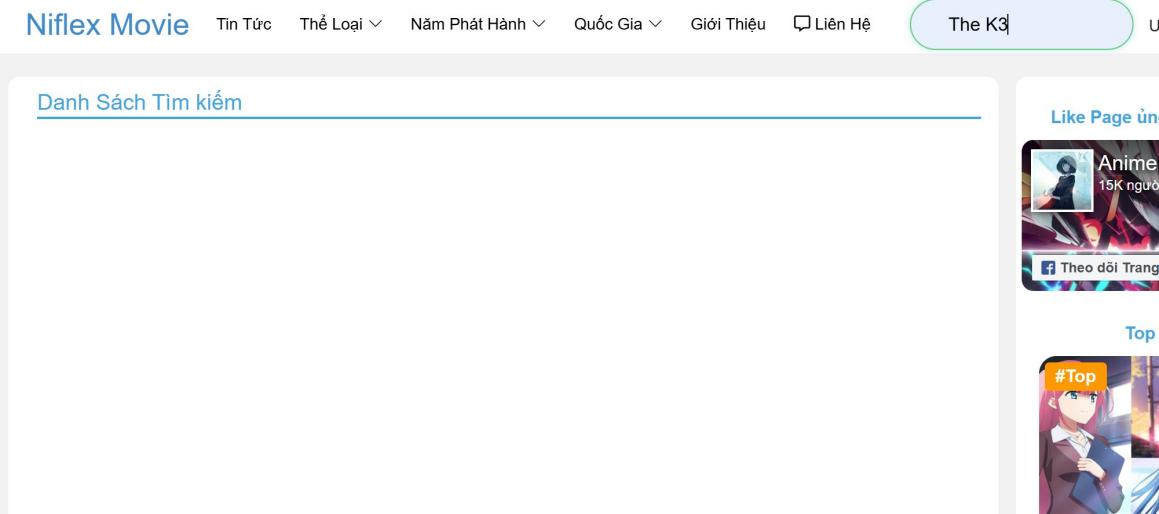
Ở chức năng tìm kiếm sau khi người dùng truy cập vào trang chủ web của phim và nhấn button tìm kiếm và nhập thông tin tên bộ phim muốn tìm. Hệ thống sẽ tiến hành kiểm tra với thông tin được lưu ở cơ sở dữ liệu nếu tìm thấy sẽ trả ra bộ phim người dùng vừa nhập tên.

Danh Sách Tìm kiếm



Hình 5-14 Giao diện tìm kiếm phim bằng tên phim (Người dùng)

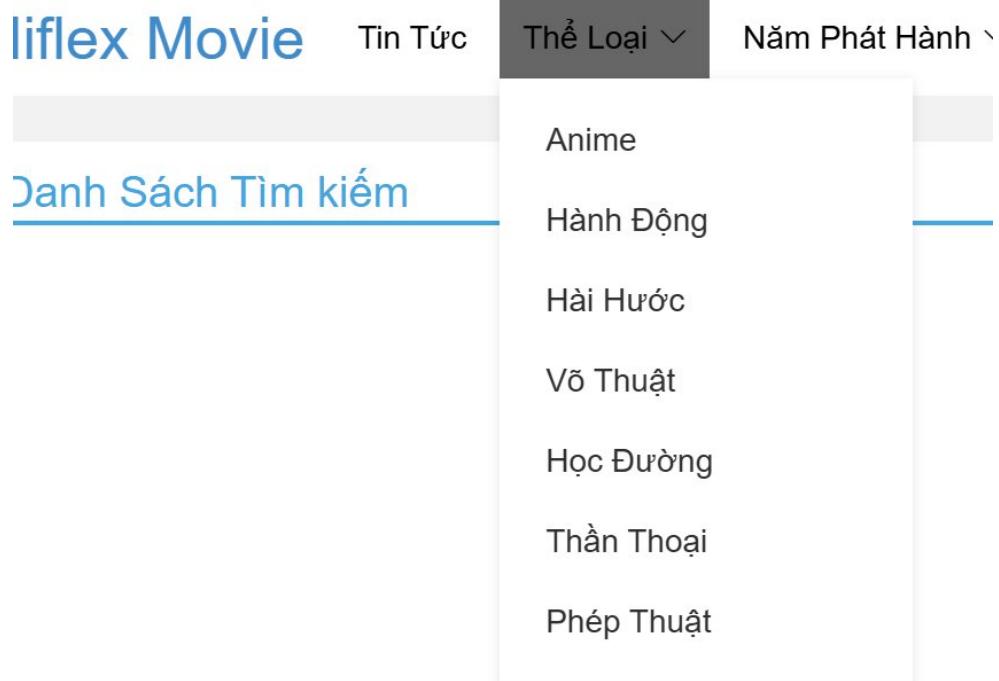
Người dùng có thể thực hiện chức năng tìm kiếm mà không cần phải đăng nhập vào trang web phim. Nếu tên phim người dùng nhập không được tìm thấy trong csdl sẽ hiển thị ra trang chủ trắng trên web.



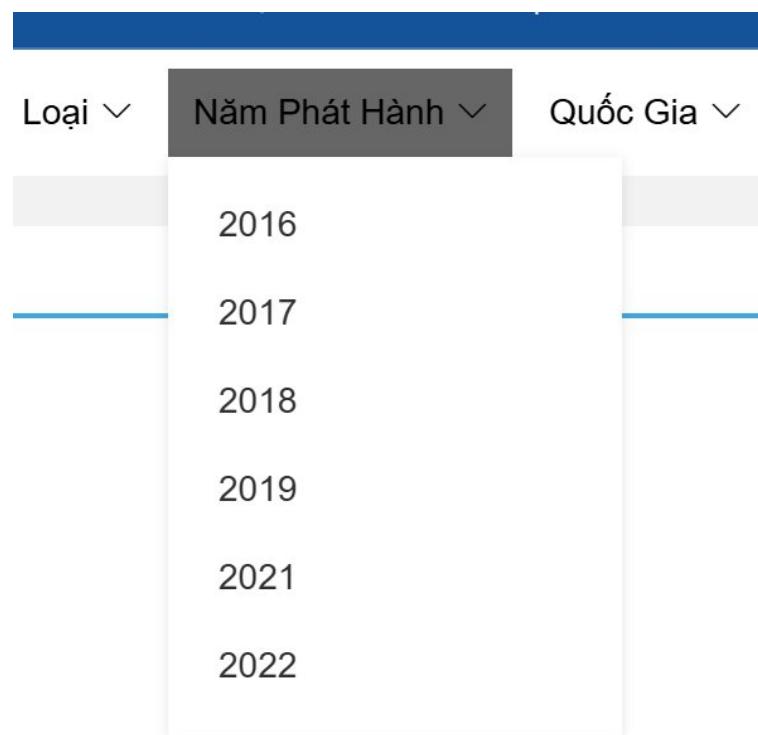
Hình 5-15 Giao diện tìm kiếm phim bằng tên phim (Người dùng)

Tìm kiếm phân loại (thể loại phim, năm phát hành, quốc gia)

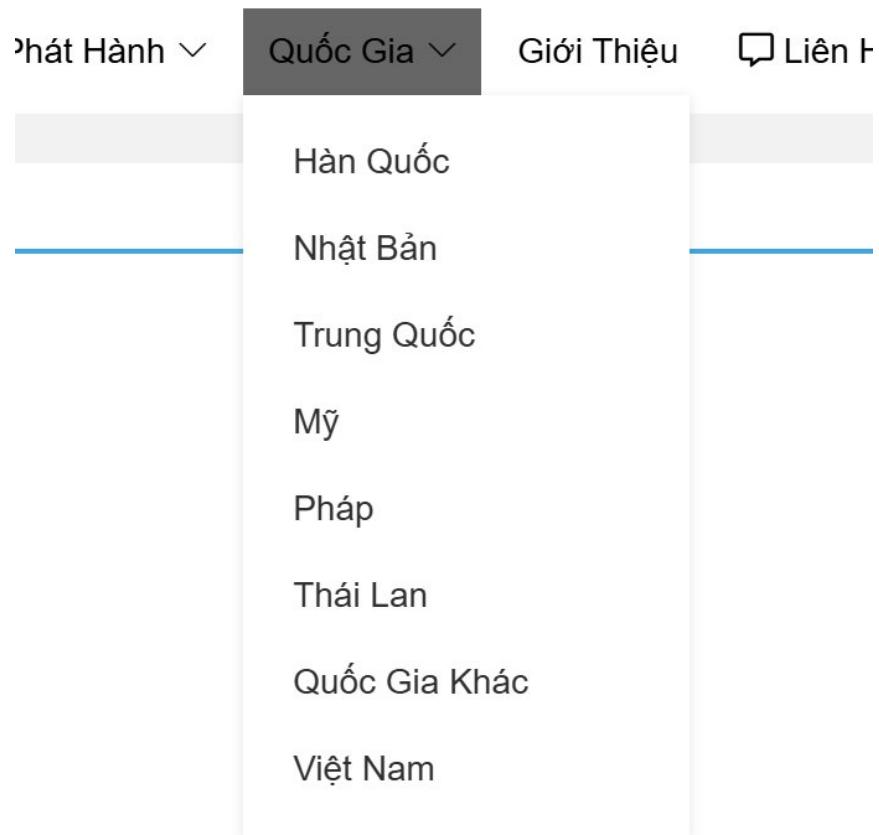
Tương tự như chức năng tìm kiếm nhưng ở thao tác tìm kiếm này, người dùng sẽ không cần người dùng nhập vào thông tin thay vào đó sẽ được chọn 1 trong số những thông tin có sẵn ở dropdown menu tùy thuộc vào thông tin người dùng muốn tìm kiếm như: Thể loại, Năm Phát Hành, Quốc Gia.



Hình 5-16 Giao diện tìm kiếm phim theo thể loại phim



Hình 5-17 Giao diện tìm kiếm phim theo năm phát hành



Hình 5-18 Giao diện tìm kiếm phim theo quốc gia

Khi người dùng click vào thông tin button hệ thống sẽ xuất ra những thông tin tương ứng.

VD1: Nếu người dùng chọn quốc gia là ‘Hàn Quốc’ ta sẽ trả về kết quả những bộ phim thuộc quốc gia Hàn Quốc.

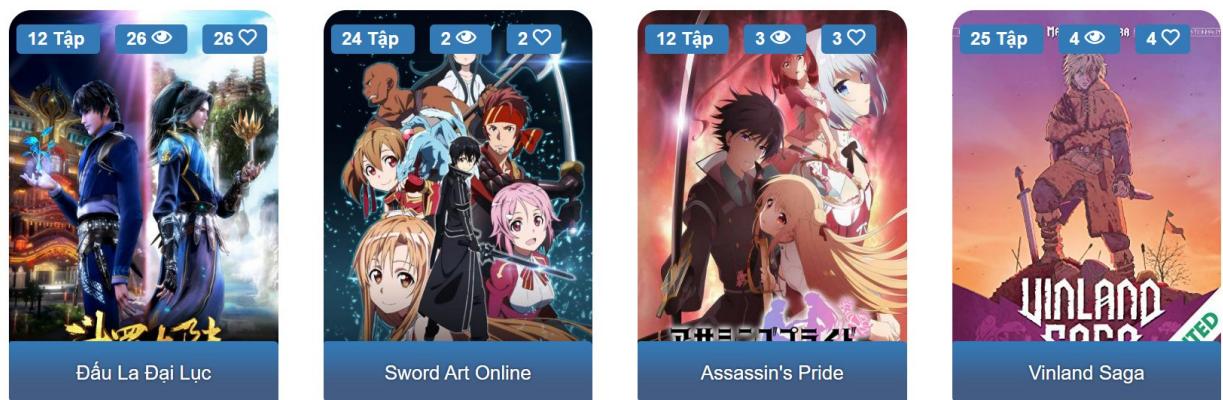
Danh sách tìm kiếm



Hình 5-19 Giao diện kết quả tìm kiếm

VD2: Nếu người dùng chọn thẻ loại là ‘Anime’ ta sẽ trả về kết quả những bộ phim thuộc thẻ loại Anime.

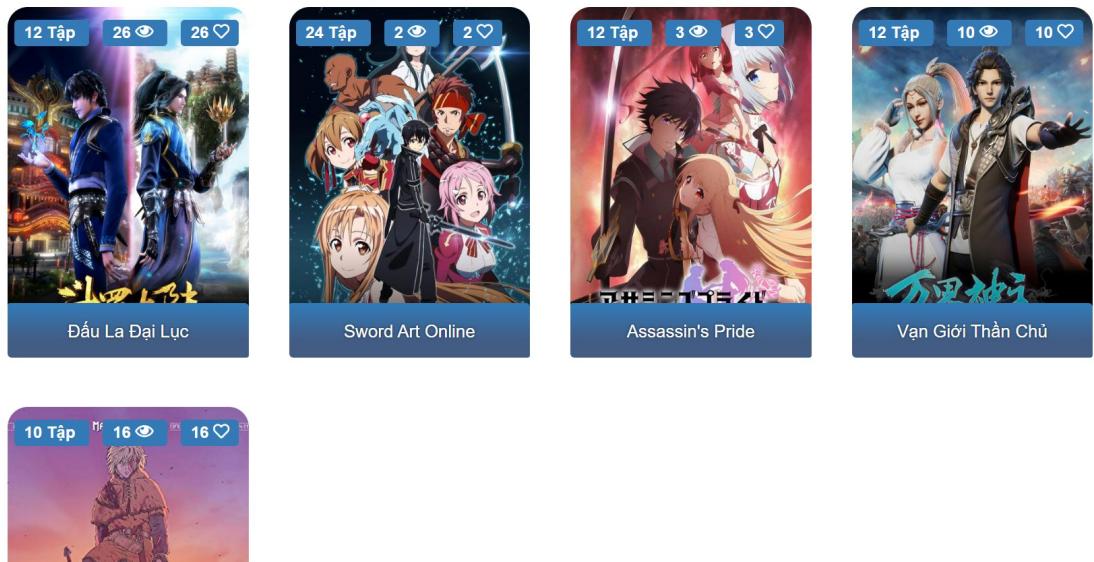
Danh Sách Tìm kiếm



Hình 5-20 Giao diện kết quả tìm kiếm

VD2: Nếu người dùng chọn Năm Phát Hành là ‘2016’ ta sẽ trả về kết quả những bộ phim được phát hành vào năm 2016

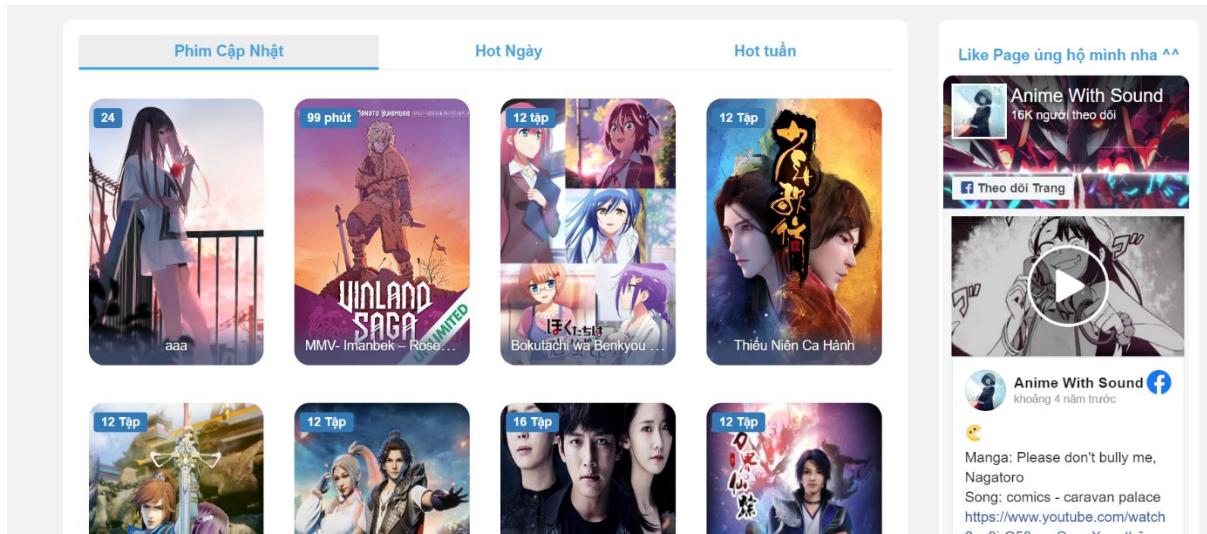
Danh Sách Tìm kiếm



Hình 5-21 Giao diện kết quả tìm kiếm

Xem phim tập, phim lẻ

Ngay tại giao diện trang chủ sẽ hiển thị danh sách được phân loại theo phim bộ và phim lẻ. Mỗi thể loại sẽ được thiết kế phân trang và tách thành hai danh sách để cho người dùng dễ dàng tìm kiếm và chọn ra bộ phim phù hợp.

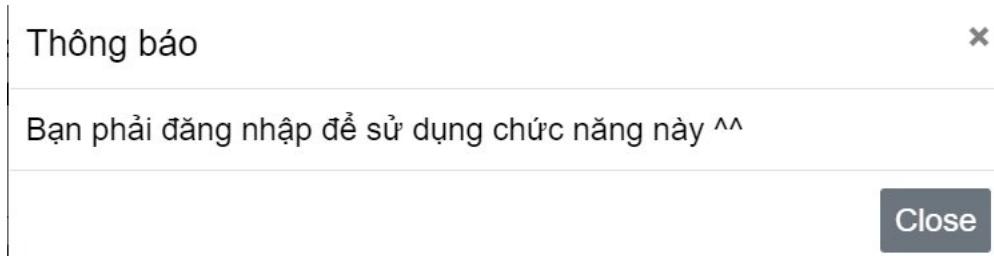


Hình 5-22 Giao diện hiển thị danh sách các phim bộ và phim lẻ ở trang chủ

Thêm phim xem sau

Để thực hiện chức năng này hệ thống người dùng sẽ phải đăng nhập trên web phim. Nếu

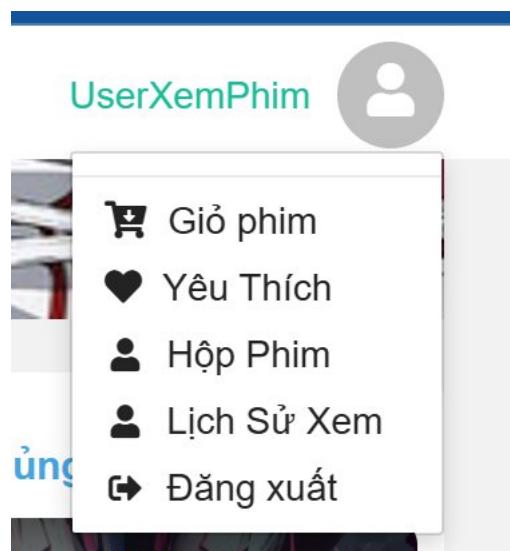
người dùng click vào button ‘Xem sau’ mà chưa thực hiện đăng nhập hệ thống sẽ hiển thị thông báo cho người dùng yêu cầu đăng nhập. Trong trường hợp người dùng chưa có tài khoản thì sẽ phải thực hiện đăng ký để sử dụng chức năng này.



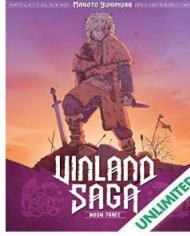
Nếu người dùng đã thực hiện xong thao tác đăng nhập sẽ hiển thị thông báo thêm vào xem sau thành công cho người dùng.



Sau đó người dùng sẽ click vào button . Ở đây người dùng sẽ thấy dropdown menu bao gồm (Giỏ phim, Yêu thích, Hộp phim, Lịch sử xem, Đăng xuất). Nếu người dùng muốn xem danh sách những bộ phim đã lưu vào xem sau. Chỉ cần click vào button ‘Hộp phim’ hệ thống sẽ trả về những bộ phim mà người dùng đã lưu.



Danh Sách Xem Sau

Mã Phim	Tên Phim	Hình	Chức Năng
17	MMV- Imanbek – RosesRemix		Xem Xóa

Hình 5-23 Giao diện hiển thị danh sách phim xem sau của người dùng

Xem lịch sử các phim đã xem

Tương tự như chức năng ‘Xem sau’ để xem lại lịch sử phim đã xem người dùng sẽ phải đăng nhập trên web phim. Nếu người dùng click vào button ‘Lịch Sử Xem’ mà chưa thực hiện đăng nhập hệ thống sẽ hiển thị thông báo cho người dùng yêu cầu đăng nhập. Trong trường hợp người dùng chưa có tài khoản thì sẽ phải thực hiện đăng ký để sử dụng chức năng này.



Nếu người dùng đã thực hiện xong thao tác đăng nhập sẽ hiển thị danh sách những bộ phim mà người dùng đã xem.

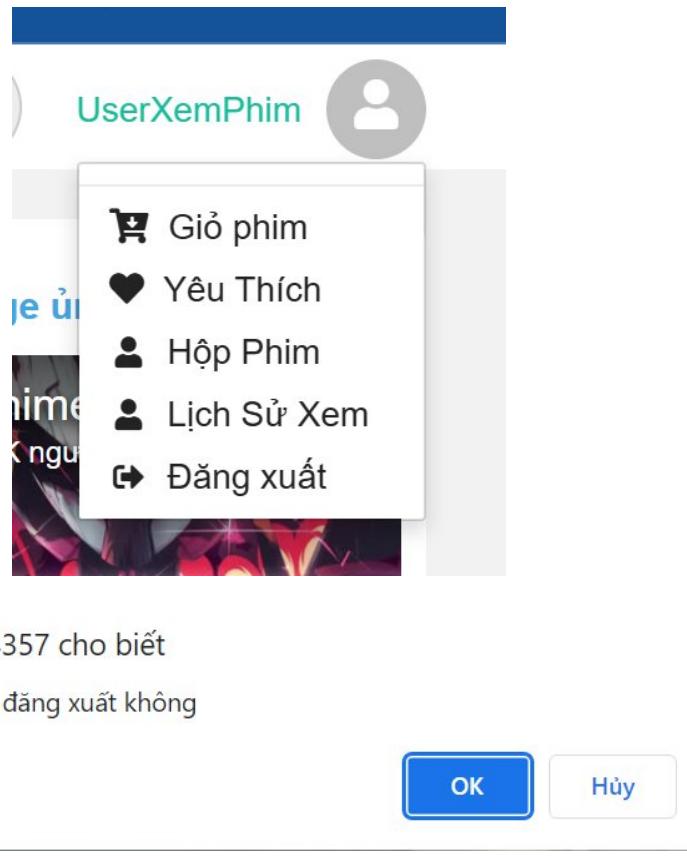
Danh Sách Xem Sau

Mã Phim	Tên Phim	Hình	Chức Năng
17	MMV- Imanbek – RosesRemix		Xem Xóa

Hình 5-24 Giao diện hiển thị các phim đã xem của người dùng (lịch sử xem phim)

Chức năng đăng xuất

Khi người dùng muốn sử dụng chức năng đăng xuất hệ thống sẽ kiểm tra người dùng đã đăng nhập chưa. Trong trường hợp người dùng chưa đăng nhập sẽ phải đăng nhập để sử dụng chức năng. Còn nếu người dùng đã đăng nhập khi đó chỉ cần click vào đăng xuất sau đó hệ thống sẽ hiển thị ra thông báo cho người dùng. Nếu người dùng nhấn ‘Ok’ sẽ được chuyển đến trang chủ phim trong trạng thái chưa đăng nhập.



Hình 5-25 Giao diện đăng xuất

Chức năng thêm phim yêu thích

Trong quá trình xem phim, nếu người dùng cảm thấy bộ phim nào hay hoặc cần lưu ý lại thông tin phim để hôm sau có thể quay lại xem tiếp hoặc giới thiệu cho bạn bè thì người dùng có thể sử dụng chức năng này. Chỉ hữu dụng chỉ khi người dùng đăng nhập tài khoản vào hệ thống. Tại giao diện chi tiết nội dung của phim thì sẽ có nút biểu tượng “Yêu thích” phim. Người dùng bấm vào nút đó thì hệ thống sẽ xử lý đưa phim đó vào danh sách yêu thích phim theo tên tài khoản đăng nhập tương ứng của người dùng.



Black Clover

Thời lượng 24 Tập
Quốc gia Nhật Bản
Thể loại Anime
Phát hành 2021
Lượt xem 26
Giá phim 25000 VND

[Yêu thích](#) [Đặt mua](#)

[Xem Phim](#)

[Xem sau](#)

Sau đó người dùng vẫn có thể xem lại được danh sách các phim đã bấm yêu thích, tất cả được lưu lại ở mục “Yêu thích” phim.

Danh Sách Lượt Yêu Thích Phim Bộ

Mã Phim	Tên Phim	Hình	Chức Năng
5	Black Clover		Xem Xóa
17	MMV- Imanbek – RosesRemix		Xem Xóa

Hình 5-26Giao diện danh sách phim đã được thêm vào yêu thích

Chức năng đặt mua phim

Tương tự như thêm phim vào yêu thích, người dùng vẫn có thể tìm các phim ưng ý theo sở thích sau đó chọn các phim đang được bán (chỉ khi mua mới được xem) để cho vào giỏ đặt phim và sau đó thanh toán phim. Người dùng có thể xem lại danh sách các phim đã đặt mua trong giỏ phim, tuy nhiên khi thanh toán người dùng chỉ được thanh toán một lúc một phim cho một phiên thanh toán mua phim.

Tính năng chỉ được sử dụng khi người đã đăng nhập tài khoản vào hệ thống xem phim trực tuyến này. Mỗi phim chỉ được mua số lượng là một cho một tài khoản người dùng. Người dùng vẫn có thể xóa thông tin phim đã đặt mua trong giỏ phim đi bằng cách nhấn vào nút “Xóa”.

Mã Phim	Tên Phim	Hình	Giá	Chức Năng
5	Black Clover		25000 VND	<button>Thanh Toán</button> <button>Xóa</button>

Thanh Toán Đơn Đặt Phim

Hình 5-27Giao diện danh sách phim đã được đặt mua đang chờ thanh toán

Quản trị viên website

Đăng nhập

Tương tự như người dùng quản trị viên cũng cần đăng nhập để hệ thống kiểm tra thông tin tài khoản. Nếu hệ thống kiểm tra thông tin là ‘Admin’ sẽ đưa người quản trị đến trang chủ quản trị

TÀI KHOẢN
6

PHIM LẺ
4

PHIM BỘ
12

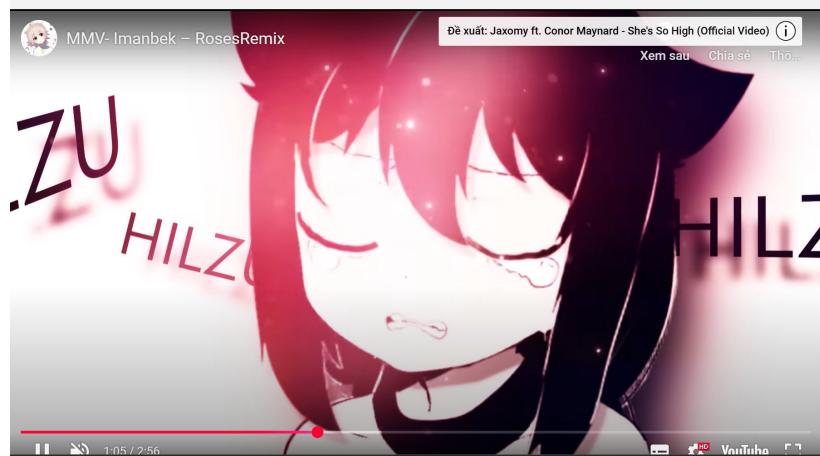
LƯỢT THÍCH
4

Xuất báo cáo

Hiển thị top phim (Những bộ phim có lượt xem nhiều nhất), phim mới cập nhật

Ở chức năng này sẽ hiển thị những bộ phim hay hấp dẫn để đề xuất cho người dùng. Dựa vào số lượt xem được thống kê trên trang chủ quản trị (theo phim bộ hoặc phim lẻ). Số lượt

xem sẽ tăng lên mỗi khi người dùng nhấn vào button ‘xem phim’ và xem



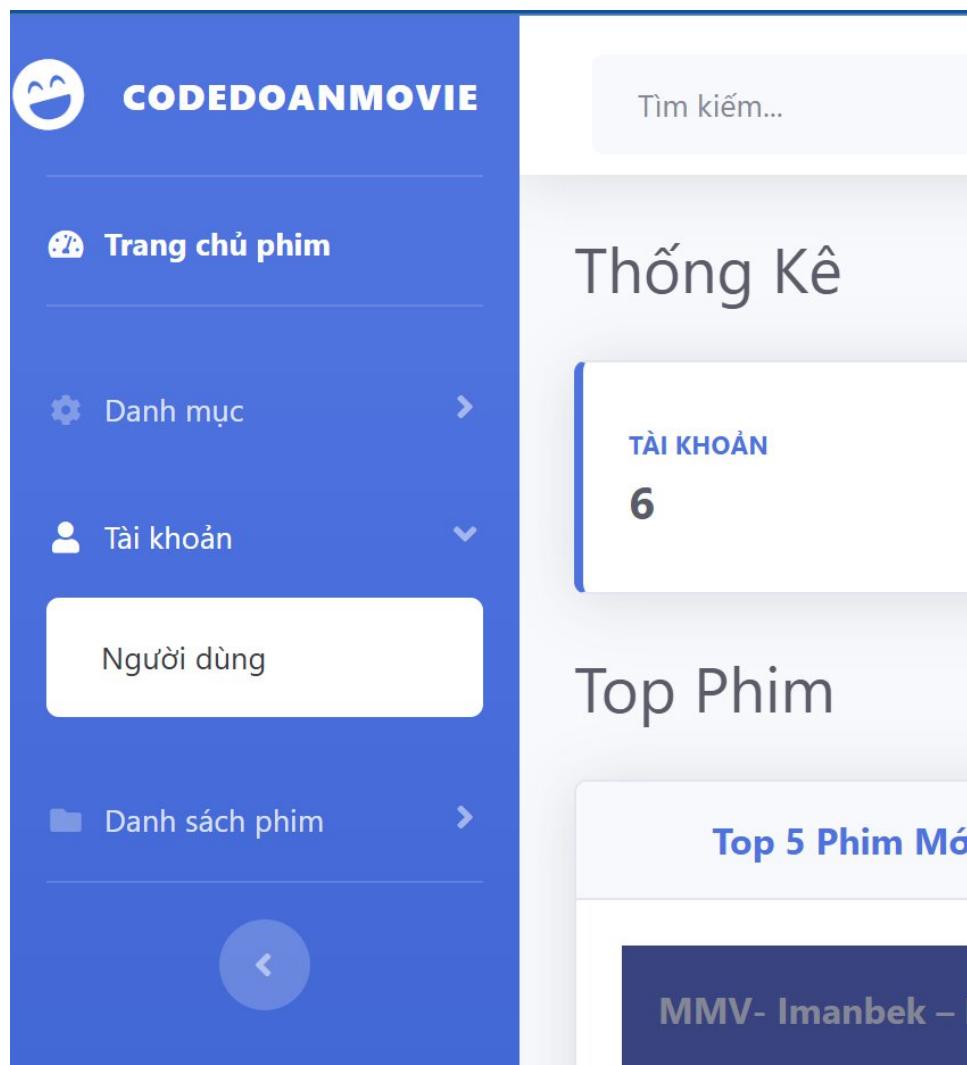
Top Phim

Phim Mới Cập Nhật		Phim Lẻ Xem Nhiều Nhất		Phim bộ Xem nhiều nhất	
aaa	24	11	1 Lượt	Bokutachi wa Benkyou ga Dekinai S2	45 Lượt
MMV- Imanbek – RosesRemix	99 phút			Đấu La Đại Lục	25 Lượt
Bokutachi wa Benkyou ga Dekinai S2	12 tập			Black Clover	25 Lượt
Thiếu Niên Ca Hành	12 Tập			Thiếu Niên Ca Hành	15 Lượt
Linh Kiếm Tôn	12 Tập			Vạn Giới Thần Chủ	9 Lượt
Vạn Giới Thần Chủ	12 Tập			MMV- Imanbek – RosesRemix	8 Lượt
The K2	16 Tập			Linh Kiếm Tôn	5 Lượt

Hình 5-28 Giao diện hiển thị thống kê top phim theo lượt xem, phim mới

Quản lý tài khoản người dùng (thêm sửa xoá)

Chức năng quản lý tài khoản người dùng được thực hiện trên quyền quản trị của ‘Admin’. Người quản trị có thể tiến hành thao tác thêm, sửa, xoá để phù hợp với nhu cầu của mình. Và chỉ có ‘Admin’ mới có quyền thay đổi người dùng khác sang thành quyền ‘Admin’. Người quản trị khi click vào button ‘Người dùng’ trong dropdown menu của trang quản trị hệ thống sẽ hiển thị danh sách những tài khoản được lưu trữ.



Khi người quản trị muốn thêm ‘tài khoản’ sẽ click vào button ‘Thêm tài khoản’ sẽ tiến hành nhập những thông tin cần thiết bao gồm: (Tên đăng nhập, Mật Khẩu, Email, Quyền).

Tài khoản				
Hiển thị <input type="button" value="5"/> dòng <input type="text" value="Tìm kiếm:"/>				
Tên đăng nhập	Mật khẩu	Quyền truy cập	Email	Chức năng
UserXemPhim1	601f1889667efaebb33b8c12572835da3f027f78	User	UserXemPhim1@gmail.com	
Tên đăng nhập	Mật khẩu	Quyền truy cập	Email	Chức năng

Hình 5-29 Giao diện quản lý danh sách tài khoản (mã hóa mật khẩu)

Khi người dùng đăng ký tài khoản và đăng nhập lại hoặc người quản trị thêm vào tài khoản mới thì tất cả mật khẩu của người dùng được mã hóa để bảo mật dữ liệu, an toàn hơn

về thông tin mật khẩu cho tài khoản người dùng.

Thêm tài khoản

Tên đăng nhập

Mật khẩu

Email

Quyền truy cập

User

Thêm

Trong trường hợp người quản trị muốn cập nhật thông tin của người dùng chỉ cần click vào icon ‘Chỉnh sửa’ sau đó tiến hành nhập cập nhật thông tin. Dữ liệu sẽ được thay đổi và lưu xuống database.

Sửa tài khoản

Tên đăng nhập UserXemPhim1

Mật khẩu 601f1889667efaebb33b8c

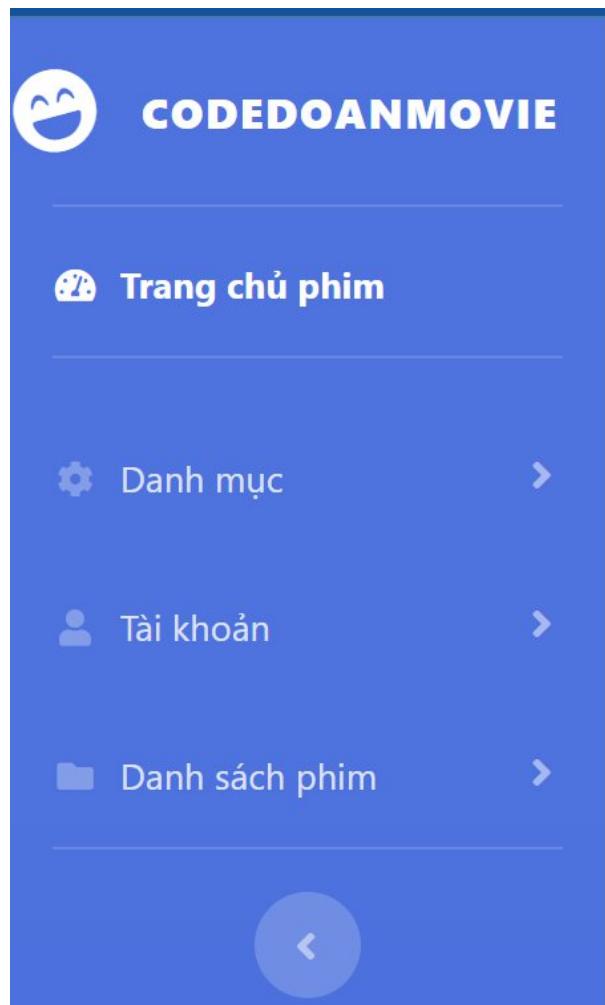
Email UserXemPhim1@gmail.co

Lưu cập nhật

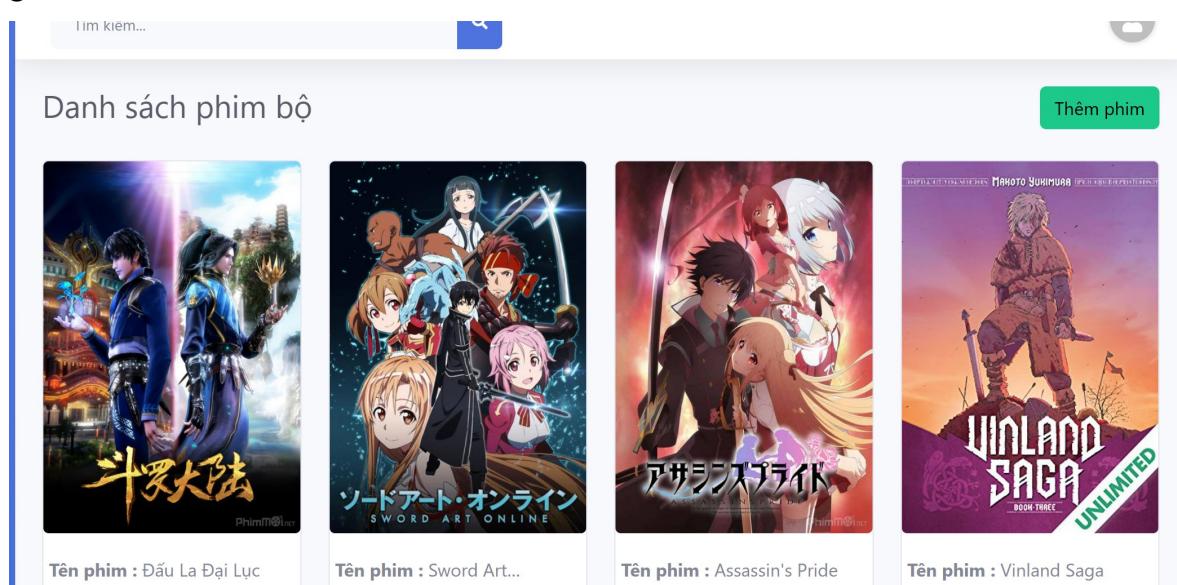
Trở lại

Quản lý danh sách phim bộ, phim lẻ

Chức năng quản lý phim cho phép người quản trị có thể thay đổi số tập phim và những thông tin liên quan đến phim.



Sau khi click vào ‘Phim bộ’ hệ thống sẽ hiển thị ra danh sách các phim bộ được lưu trữ trong cơ sở dữ liệu.



The screenshot displays a list of movies. At the top is a search bar with placeholder text 'Tim kiếm...' and a magnifying glass icon. Below the search bar is a section titled 'Danh sách phim bộ'. Four movie posters are shown in a row:

- Tên phim :** Đấu La Đại Lục
- Tên phim :** Sword Art Online
- Tên phim :** Assassin's Pride
- Tên phim :** Vinland Saga

A green button labeled 'Thêm phim' is located in the top right corner of the movie list section.

Hình 5-30 Giao diện quản lý danh sách các phim bộ

Khi người quản trị muốn thêm một bộ phim mới chỉ cần click vào button ‘Thêm phim bộ’ sau đó nhập thông tin muốn thêm vào và nhấn vào button ‘Lưu cập nhật’. Sau khi click phim sẽ được thêm vào cơ sở dữ liệu.

Thêm Phim Bộ

Tên Phim

Nội dung

Năm Phát Hành

2016

Thể Loại

Anime

Thời Lượng

Hình No file chosen

Trong trường hợp người quản trị muốn cập nhật thông tin phim chỉ cần click vào icon ‘Chỉnh sửa’ sau đó tiến hành nhập cập nhật thông tin và nhấn vào button ‘Lưu cập nhật’. Dữ liệu sẽ được thay đổi và lưu xuống database.

Sửa Phim



No file chosen

Tên Phim

Sword Art Online

Nội dung

Con đường sống duy nhất là đánh bại mọi kẻ thù. Cái chết trong game đồng nghĩa với

Năm phát hành

2016

Thể loại

Anime

Quốc gia

Nhật Bản

Thời lượng

24 Tập

Trường hợp người quản trị muốn xoá phim ra khỏi danh mục hệ thống sẽ hiển thị thông báo đến để yêu cầu xác nhận về việc xoá phim. Sau khi đồng ý với yêu cầu trên. Phim sẽ được xoá khỏi cơ sở dữ liệu.

localhost:44357 cho biết

Bạn có chắc muốn xóa phim này không?

OK

Hủy

Nếu người quản trị muốn thêm một tập phim mới vào bộ phim chỉ cần click vào button ‘Thêm Tập’ và điền link phim cần thiết sau khi bấm ‘Thêm’ sẽ được lưu vào cơ sở dữ liệu.

The screenshot shows two movie entries in a list:

Tên phim	Nội dung	Thể Loại	Số Tập	Năm Phát Hành	Giá phim
Đấu La Đại Lục	Một đại lục không hề yên bình, một cuộc sống đầy hiểm nguy...	Anime	12 Tập	2016	20000 VND
Sword Art Online	Con đường sống duy nhất là đánh bại mọi kẻ thù. Cái chết trong game...	Anime	24 Tập	2016	22000 VND

Each entry has three buttons at the bottom: 'Thêm tập' (Add episode), 'Sửa thông tin' (Edit information), and a red trash can icon.

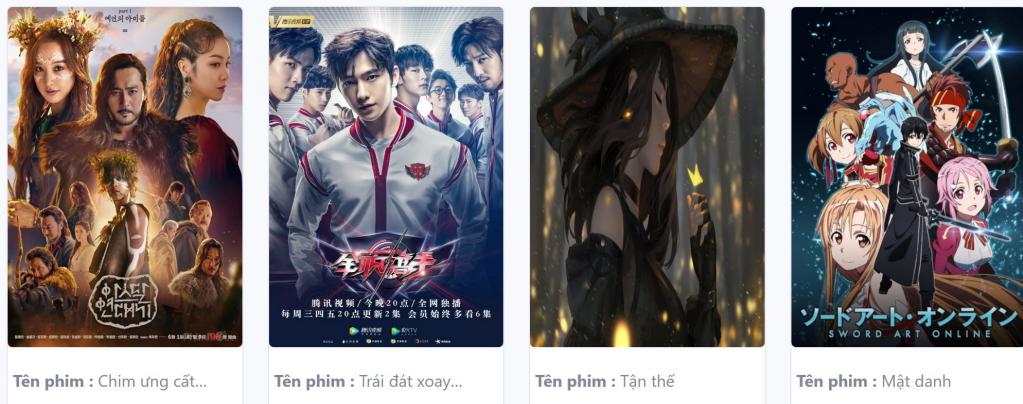
The screenshot shows the 'Thêm tập' (Add episode) form for the movie 'Sword Art Online':

Tên Phim : Sword Art Online
Thêm tập : 2
Link Tập Phim
[Input field]
[Thêm] (Add) button
[Trở lại] (Back) button

Tương tự như phim bộ sau khi click vào ‘Phim lẻ’ hệ thống sẽ hiển thị ra danh sách các phim lẻ được lưu trữ trong cơ sở dữ liệu.

Danh sách phim lẻ

Thêm phim



Hình 5-31 Giao diện quản lý danh sách các phim lẻ

Khi người quản trị muốn thêm một bộ phim mới chỉ cần click vào button 'Thêm phim' sau đó nhập thông tin muốn thêm vào và nhấn vào button 'Lưu cập nhật'. Sau khi click phim sẽ được thêm vào cơ sở dữ liệu.

Thêm Phim Lẻ

Tên Phim

Nội dung

Năm Phát Hành

 ▼

Thể Loại

 ▼

Thời Lượng

Hình No file chosen

Trong trường hợp người quản trị muốn cập nhật thông tin phim chỉ cần click vào icon 'Chỉnh sửa' sau đó tiến hành nhập cập nhật thông tin và nhấn vào button 'Lưu cập nhật'. Dữ liệu sẽ được thay đổi và lưu xuống database.

Quản lý danh mục (tin tức, banner quảng bá phim, thể loại phim, quốc gia, năm phát hành, nội dung giới thiệu website)



Để tiến hành chỉnh sửa người quản trị sẽ click vào những button tương ứng để tiến hành việc chỉnh sửa phù hợp với nhu cầu. Trong trường hợp người quản trị muốn cập nhật thông tin chỉ cần click vào icon ‘Chỉnh sửa’ sau đó tiến hành nhập cập nhật thông tin. Dữ liệu sẽ được thay đổi và lưu xuống database. Và hiển thị lên web phim để tiếp cận đến người sử dụng.

Tin tức	Thời gian	Thao tác
Phim sắp cập nhật	10/1/2022 12:00:00 AM	Sửa Xóa
Chưa nghỉ ra	9/30/2022 12:00:00 AM	Sửa Xóa
Thời gian bảo trì	10/1/2022 12:00:00 AM	Sửa Xóa

Hình 5-32 Giao diện quản lý danh sách các tin tức của phim

Sau khi điền đầy đủ thông tin cần thiết người quản trị sẽ nhấn vào button ‘Thêm’ khi đó những thông tin được nhập sẽ được lưu xuống cơ sở dữ liệu.

Thêm Tin Tức

Tiêu đề

Tóm tắt

Nội dung

Hình No file chosen

Nếu người dùng muốn thay đổi tin tức chỉ cần click vào button ‘Sửa’ sau đó tiến hành thay đổi thông tin và click vào button ‘Lưu cập nhật’ để lưu thông tin xuống database.

Sửa Tin Tức



Tiêu đề
Phim sắp cập nhật:

Tóm tắt
CHIẾN CÔNG LỊCH SỬ CỦA 24 AE VIỆT NAM - RAID TOANG NHÀ PHILIPPINES !!! - SCU

Nội dung
CHIẾN CÔNG LỊCH SỬ CỦA 24 AE VIỆT NAM - RAID TOANG NHÀ PHILIPPINES !!! - SCU

Ngày cập nhật
10/1/2022 12:00:00 AM

Không có tệp nào được chọn

Trường hợp người quản trị muốn xoá tin tức ra khỏi danh mục hệ thống sẽ hiển thị thông báo đến để yêu cầu xác nhận về việc xoá tin tức. Sau khi đồng ý với yêu cầu trên. Tin tức sẽ được xoá khỏi cơ sở dữ liệu.

localhost:44357 cho biết

Bạn có chắc muốn xoá tin tức này không ?

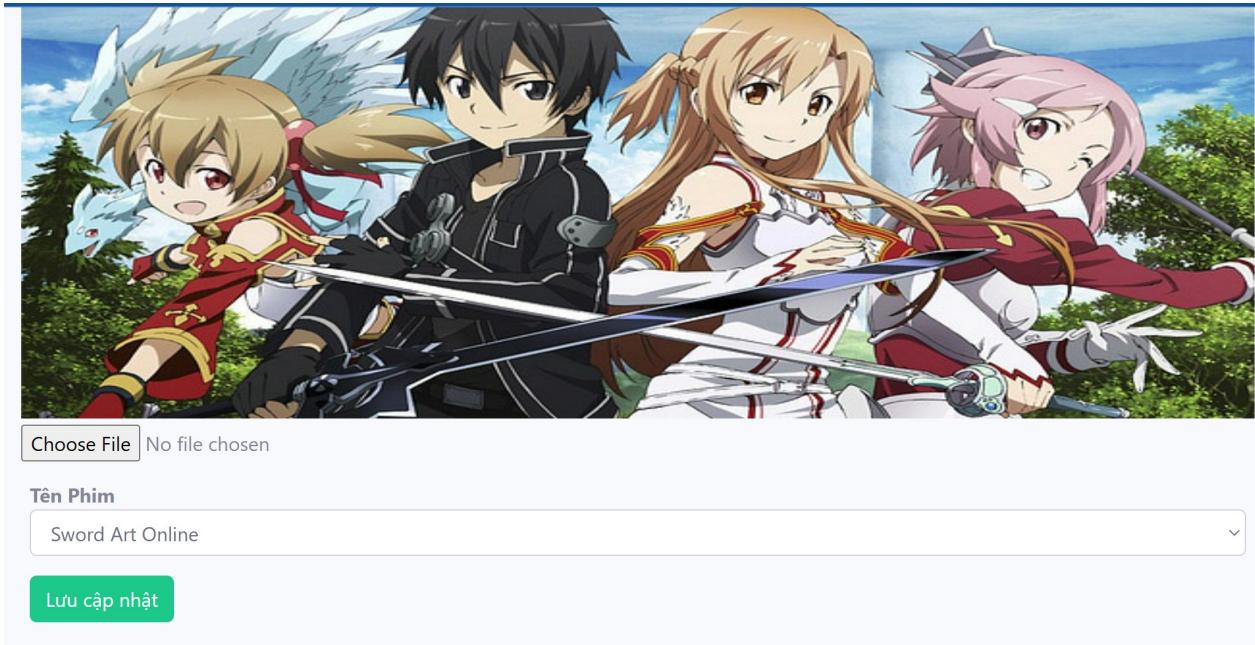
Tương tự như những thông tin trên khi người quản trị muốn thêm một banner xuất hiện trên web phim sẽ click vào button ‘Thêm Banner’ và cập nhật thông tin cho banner đó, sau khi bấm vào button ‘Lưu cập nhật’ banner sẽ được thêm vào cơ sở dữ liệu.

Mã Banner	Hình	Phim	Chức năng
2		Sword Art Online	 
3		Black Clover	 

Hiển thị 3 trong 3

Trước 1 Sau

Hình 5-33 Giao diện quản lý danh sách các hình ảnh banner quảng cáo của phim



Trường hợp người quản trị muốn xoá banner ra khỏi danh mục hệ thống sẽ hiển thị thông báo đến để yêu cầu xác nhận về việc xoá Banner. Sau khi đồng ý với yêu cầu trên. Banner sẽ được xoá khỏi cơ sở dữ liệu.

localhost:44357 cho biết

Bạn có chắc muốn xóa banner này không ?

OK

Hủy

Sau khi click vào ‘Quốc gia’ hệ thống sẽ hiển thị ra danh sách các quốc gia được lưu trữ trong cơ sở dữ liệu.

Quốc gia		
Hiển thị: 5 dòng	Tim kiếm:	
Mã quốc gia	Tên quốc gia	Chức năng
1	Hàn Quốc	[Edit] [Delete]
2	Nhật Bản	[Edit] [Delete]
3	Trung Quốc	[Edit] [Delete]
4	Mỹ	[Edit] [Delete]

Hình 5-34 Giao diện quản lý danh sách các quốc gia sản xuất phim

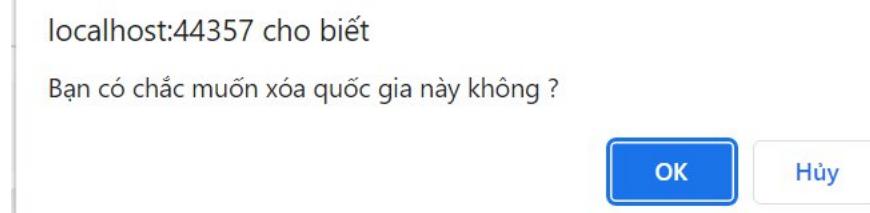
Khi người quản trị muốn thêm một quốc gia mới chỉ cần click vào button ‘Thêm quốc gia’ sau đó nhập tên quốc gia muốn thêm vào và nhấn vào button ‘Lưu cập nhật’. Sau khi click quốc gia sẽ được thêm vào cơ sở dữ liệu.

Thêm quốc gia

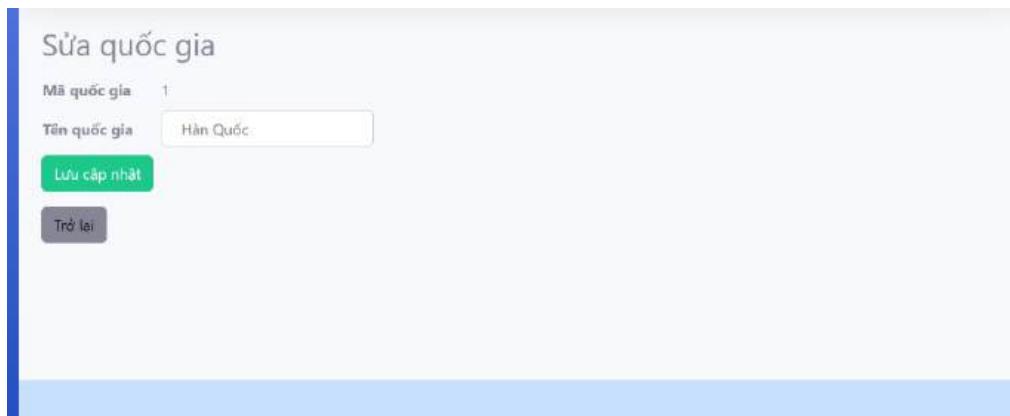
Tên quốc gia:

Thêm **Trở lại**

Trường hợp người quản trị muốn xoá quốc gia ra khỏi danh mục hệ thống sẽ hiển thị thông báo đến để yêu cầu xác nhận về việc xoá quốc gia. Sau khi đồng ý với yêu cầu trên. Quốc gia sẽ được xoá khỏi cơ sở dữ liệu.



Trong trường hợp người quản trị muốn cập nhật thông tin quốc gia chỉ cần click vào icon ‘Chỉnh sửa’ sau đó tiến hành nhập cập nhật thông tin. Dữ liệu sẽ được thay đổi và lưu xuống database khi nhấn vào button ‘Lưu cập nhật’.



Sau khi click vào ‘Thể Loại’ hệ thống sẽ hiển thị ra danh sách các thể loại được lưu trữ trong cơ sở dữ liệu.

Danh sách		
Hiển thị	đóng	Tìm kiếm:
Mã thể loại	Tên thể loại	Chức năng
1	Anime	
2	Hành Động	
3	Hài hước	
4	Võ thuật	
5	Học đường	

Hình 5-35Giao diện quản lý danh sách thể loại phim

Khi người quản trị muốn thêm thể loại mới chỉ cần click vào button ‘Thêm thể loại’ sau đó nhập tên thể loại muốn thêm vào và nhấn vào button ‘Lưu cập nhật’. Sau khi click thể loại sẽ được thêm vào cơ sở dữ liệu.

Thêm Thể loại

Tên thể loại

Thêm

Trở lại

Trong trường hợp người quản trị muốn cập nhật thông tin thẻ loại chỉ cần click vào icon ‘Chỉnh sửa’ sau đó tiến hành nhập cập nhật thông tin. Dữ liệu sẽ được thay đổi và lưu xuống database khi nhấn vào button ‘Lưu cập nhật’.

Trường hợp người quản trị muốn xoá thẻ loại ra khỏi danh mục hệ thống sẽ hiển thị thông báo đến để yêu cầu xác nhận về việc xoá thẻ loại. Sau khi đồng ý với yêu cầu trên. Thẻ loại sẽ được xoá khỏi cơ sở dữ liệu.

localhost:44357 cho biết
Bạn có chắc muốn xóa thẻ loại này không ?

OK

Hủy

Sau khi click vào ‘Năm phát hành’ hệ thống sẽ hiển thị những thông tin được lưu trữ trong cơ sở dữ liệu.

Mã Năm	Tên Năm	Chức năng
1	2016	 
2	2017	 
3	2018	 
4	2019	 
5	2021	 
Mã Năm	Tên Năm	Chức năng

Hình 5-36 Giao diện quản lý danh sách năm phát hành các phim

Khi người quản trị muốn thêm năm phát hành mới chỉ cần click vào button ‘Thêm năm phát hành’ sau đó nhập năm phát hành muốn thêm vào và nhấn vào button ‘Lưu cập nhật’. Sau khi click năm phát hành sẽ được thêm vào cơ sở dữ liệu

Thêm Năm Phát Hành

Năm

Thêm

Trở lại

Trong trường hợp người quản trị muốn cập nhật thông tin năm phát hành chỉ cần click vào icon ‘Chỉnh sửa’ sau đó tiến hành nhập cập nhật thông tin. Dữ liệu sẽ được thay đổi và lưu xuống database khi nhấn vào button ‘Lưu cập nhật’.

Trường hợp người quản trị muốn xoá năm phát hành ra khỏi danh mục hệ thống sẽ hiển thị thông báo đến để yêu cầu xác nhận về việc xoá năm phát hành. Sau khi đồng ý với yêu cầu trên. Năm phát hành sẽ được xoá khỏi cơ sở dữ liệu.

localhost:44357 cho biết

Bạn có chắc muốn xoá năm phát hành này không ?

OK

Hủy

Sau khi click vào ‘Giới thiệu’ hệ thống sẽ hiển thị những thông tin được lưu trữ trong cơ sở dữ liệu.

Thống kê số lượng tài khoản, số lượng phim bộ, phim lẻ, phim xem nhiều nhất, phim mới cập nhật.

Dựa vào dữ liệu được truy xuất từ cơ sở dữ liệu người quản trị lấy được những thông tin liên quan đến tài khoản, phim lẻ, phim bộ.

The screenshot displays the main dashboard of the CODEDOANMOVIE application. At the top left is the logo 'CODEDOANMOVIE'. To its right is a search bar with the placeholder 'Tim kiem...' and a magnifying glass icon. Further right is a user profile icon. On the far right, there is a blue button labeled 'Xuất báo cáo' (Export Report) with a document icon.

The main content area is titled 'Thống Kê' (Statistics). It features four summary cards:

- TÀI KHOẢN: 6
- PHIM LẺ: 4
- PHIM BỘ: 12
- LƯỢT THÍCH: 4

Below the statistics is a section titled 'Top Phim' (Top Movies), which is divided into three sub-sections:

- Top 5 Phim Mới Cập Nhật**
 - MMV- Imanbek – RosesRemix
10 Tập
 - Bokutachi wa Benkyou ga Dekinai S2
12 Tập
- Top 5 Phim Lẻ Xem Nhiều Nhất**
 - Chim ưng cất cánh
4 Lượt
 - Trái đất xoay vòng
3 Lượt
- Top 5 Phim Bộ Xem Nhiều Nhất**
 - Bokutachi wa Benkyou ga Dekinai S2
45 Lượt
 - Black Clover
27 Lượt

Hình 5-37 Giao diện quản lý thống kê tổng quát về phim

6. HIỆN THỰC CHƯƠNG TRÌNH

6.1 Áp dụng mẫu Facade

6.1.1. Chức năng quản lý tài khoản

❖ Code cũ của phần quản lý tài khoản, đặc biệt là phần "nhận lệnh" (Controller) có tên là TaiKhoanController, có một số chỗ chưa tốt:

- ~ **Một mình làm quá nhiều việc:** Giống như một người vừa phải dọn nhà, vừa phải nấu ăn, vừa phải sửa điện, rất vất vả và dễ sai sót. Code cũ vừa phải:
 - Nhận yêu cầu từ người dùng (ví dụ: "thêm tài khoản mới").
 - Kiểm tra xem thông tin người dùng nhập vào có đúng không (ví dụ: có bỏ trống ô nào không).
 - Đi vào "kho" (cơ sở dữ liệu) để xem có ai trùng tên đăng nhập không.
 - Tự tay "làm" tài khoản mới (ví dụ: xáo trộn mật khẩu cho an toàn).
 - Cất tài khoản mới vào "kho".
- ~ **Phụ thuộc quá nhiều vào "cái kho":** Nếu sau này chúng ta muốn đổi "kho" (ví dụ: chuyển sang một loại kho hiện đại hơn), chúng ta phải sửa *rất nhiều* chỗ trong code.
- ~ **Khó kiểm tra:** Muốn kiểm tra xem code có chạy đúng không, chúng ta *bắt buộc* phải có "kho" thật, rất mất công.
- ~ **Khó sửa, khó thêm tính năng:** Nếu muốn thay đổi cách kiểm tra trùng tên, hoặc thêm một bước "gửi email xác nhận", chúng ta phải sửa code ở nhiều nơi, dễ bị nhầm lẫn và làm hỏng code.

❖ Giải pháp: Dùng "người quản lý" (Facade)

- ~ Để giải quyết các vấn đề trên, chúng ta sẽ thuê một "người quản lý" (Facade). Người quản lý này sẽ:
 - Nhận yêu cầu từ chúng ta (ví dụ: "thêm tài khoản mới").
 - Tự mình vào "kho", kiểm tra, làm mọi thứ cần thiết.
 - Báo cáo lại kết quả cho chúng ta.

- ~ Chúng ta (Controller) sẽ không cần phải tự tay làm mọi thứ nữa, chỉ cần "sai việc" cho người quản lý.
- ❖ Đây là code đầy đủ của phần "nhận lệnh" (TaiKhoanController) *trước khi* thuê "người quản lý":

```
// Đây là code cũ, chưa tốt

using System.Web.Mvc;

using System;

using WebsiteMovie_DAN.Common;

using WebsiteMovie_DAN.Models;

public class TaiKhoanController : Controller
{
    // "Kho" được để ngay trong phòng làm việc

    private WebmovieDataContext data = new WebmovieDataContext();

    // ... (Các phần khác: xem danh sách tài khoản, xóa tài khoản)

    [HttpPost] // Khi người dùng bấm nút "Thêm"

    public ActionResult ThemTK(TaiKhoanDTO tk)
    {
        // Kiểm tra xem người dùng có bỏ trống ô nào không

        if (string.IsNullOrEmpty(tk.TenDN))
            ViewData["Loi"] = "Tên đăng nhập không được để trống !";
    }
}
```

```
// ... (Kiểm tra các ô khác)
```

```
// Tự đi vào "kho" để kiểm tra xem có ai trùng tên không
```

```
var taikhoan = data.TaiKhoans.ToList();
```

```
bool trungTenDangNhap = false;
```

```
foreach (var item in taikhoan)
```

```
{
```

```
    if (item.TenDN == tk.TenDN)
```

```
        trungTenDangNhap = true;
```

```
}
```

```
if (trungTenDangNhap)
```

```
{
```

```
    ViewData["Loi2"] = "Đã có tài khoản này";
```

```
}
```

```
else
```

```
{
```

```
    // Tự tay "làm" tài khoản mới
```

```
TaiKhoan taiKhoan = new TaiKhoan();
```

```
taiKhoan.TenDN = tk.TenDN;
```

```
taiKhoan.MatKhau = SHA_Hash.SHA1(tk.MatKhau); // Tự xáo trộn mật khẩu
```

```
taiKhoan.Email = tk.Email;
```

```
taiKhoan.Quyen = tk.Quyen == null || tk.Quyen == "False" ? false : true;

// Tự cắt tài khoản mới vào "kho"

data.TaiKhoans.InsertOnSubmit(taiKhoan);

data.SubmitChanges();

return RedirectToAction("DSTaiKhoan");

}

return View();

}

public ActionResult SuaTK(string tendn)

{

var tk = data.TaiKhoans.First(n => n.TenDN == tendn);

if (tk == null)

{

Response.SubStatusCode = 404;

return null;

}

return View(tk);

}

[HttpPost]
```

```
public ActionResult SuaTK(string tendn, FormCollection collection)
```

```
{
```

```
    var tk = data.TaiKhoans.First(n => n.TenDN == tendn);
```

```
    var mk = collection["MatKhau"];
```

```
    var email = collection["Email"];
```

```
    if (String.IsNullOrEmpty(mk))
```

```
{
```

```
        ViewData["Loi"] = "Không được để trống";
```

```
}
```

```
else
```

```
{
```

```
    tk.MatKhau = SHA_Hash.SHA1(mk); // (13)
```

```
    tk.Email = email;
```

```
    UpdateModel(tk);
```

```
    data.SubmitChanges();
```

```
    return RedirectToAction("DSTaiKhoan");
```

```
}
```

```
    return this.SuaTK(tendn);
```

```
}
```

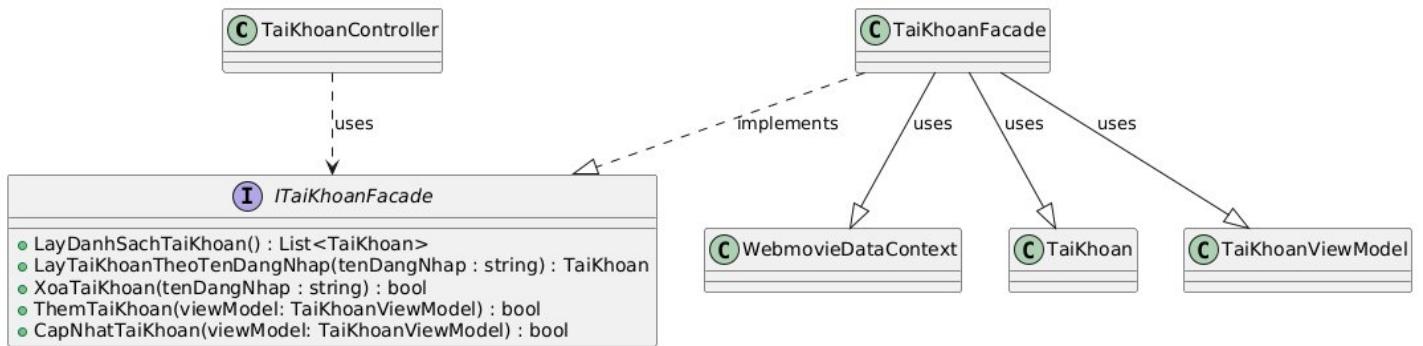
```
// ... (Phần sửa tài khoản)
```

```
}
```

~ Giải thích

- Code này làm quá nhiều việc: vừa kiểm tra thông tin, vừa vào "kho", vừa kiểm tra trùng, vừa "làm" tài khoản, vừa cất vào "kho".
- Nếu muốn thay đổi "kho", phải sửa rất nhiều chỗ trong code này.
- Muốn kiểm tra code này, phải có "kho" thật.

❖ Sơ đồ:



Hình 6-1 Sơ đồ quản lý tài khoản

❖ "Bản hợp đồng" (Interface ITaiKhoanFacade)

~ Đây là "bản hợp đồng" quy định những việc mà "người quản lý" phải làm được:

```
// ITaiKhoanFacade.cs
```

```
using System.Collections.Generic;
```

```
public interface ITaiKhoanFacade
```

```
{
```

```
    // Lấy danh sách tất cả tài khoản
```

```
    List<TaiKhoan> LayDanhSachTaiKhoan();
```

```
    // Tìm tài khoản theo tên đăng nhập
```

```
    TaiKhoan LayTaiKhoanTheoTenDangNhap(string tenDangNhap);
```

```

// Xóa tài khoản

bool XoaTaiKhoan(string tenDangNhap);

// Thêm tài khoản mới

bool ThemTaiKhoan(TaiKhoanViewModel viewModel);

// Cập nhật tài khoản

bool CapNhatTaiKhoan(TaiKhoanViewModel viewModel);

}

```

❖ "Người quản lý" (TaiKhoanFacade)

~ Đây là code của "người quản lý", làm những công việc mà "bản hợp đồng" yêu cầu:

```

// TaiKhoanFacade.cs

using System.Collections.Generic;
using WebsiteMovie_DAN.Common;
using WebsiteMovie_DAN.Facades;
using WebsiteMovie_DAN.Models;

public class TaiKhoanFacade : ITaiKhoanFacade
{
    // "Người quản lý" có chìa khóa riêng để vào "kho"
    private readonly WebmovieDataContext _dataContext;

    public TaiKhoanFacade(WebmovieDataContext dataContext)
    {
        _dataContext = dataContext;
    }

    // Hàm chuyển đổi
    private TaiKhoan MapViewModelToEntity(TaiKhoanViewModel viewModel)
    {

```

```

        return new TaiKhoan
    {
        TenDN = viewModel.TenDN,
        MatKhau = SHA_Hash.SHA1(viewModel.MatKhau), // Mã hóa ở đây
        Email = viewModel.Email,
        Quyen = viewModel.Quyen
    };
}

private TaiKhoanViewModel MapEntityToViewModel(TaiKhoan taiKhoan)
{
    return new TaiKhoanViewModel
    {
        TenDN = taiKhoan.TenDN,
        // Không bao giờ trả về mật khẩu đã mã hóa
        Email = taiKhoan.Email,
        Quyen = taiKhoan.Quyen
    };
}

// Lấy danh sách tài khoản
public List<TaiKhoan> LayDanhSachTaiKhoan()
{
    return _dataContext.TaiKhoans.ToList(); // Vào "kho" lấy danh sách
}

// Tìm tài khoản theo tên
public TaiKhoan LayTaiKhoanTheoTenDangNhap(string tenDangNhap)
{
    return _dataContext.TaiKhoans.SingleOrDefault(tk => tk.TenDN == tenDangNhap); // Vào "kho" tìm
}

```

```
// Xóa tài khoản
public bool XoaTaiKhoan(string tenDangNhap)
{
    // Tìm tài khoản cần xóa
    var taiKhoan = LayTaiKhoanTheoTenDangNhap(tenDangNhap);
    if (taiKhoan != null)
    {
        // Xóa khỏi "kho"
        _dataContext.TaiKhoans.DeleteOnSubmit(taiKhoan);
        try
        {
            _dataContext.SubmitChanges(); // Cắt thay đổi vào "kho"
            return true;
        }
        catch
        {
            return false;
        }
    }
    return false;
}

// Thêm tài khoản mới
public bool ThemTaiKhoan(TaiKhoanViewModel viewModel)
{
    // Kiểm tra xem có ai trùng tên không
    if (_dataContext.TaiKhoans.Any(tk => tk.TenDN == viewModel.TenDN))
    {
        return false; // Không thêm nếu trùng
    }

    // "Làm" tài khoản mới
```

```
var taiKhoan = MapViewModelToEntity(viewModel);

// Thêm vào "kho"
_dataContext.TaiKhoans.InsertOnSubmit(taiKhoan);

try
{
    _dataContext.SubmitChanges(); // Cắt thay đổi vào "kho"
    return true;
}
catch
{
    return false;
}

public bool CapNhatTaiKhoan(TaiKhoanViewModel viewModel)
{
    var existingTaiKhoan = LayTaiKhoanTheoTenDangNhap(viewModel.TenDN);
    if (existingTaiKhoan != null)
    {
        //Chỉ cập nhật các trường cho phép
        existingTaiKhoan.Email = viewModel.Email;
        existingTaiKhoan.MatKhau = SHA_Hash.SHA1(viewModel.MatKhau); // Mã hóa
        mật khẩu mới

        try
        {
            _dataContext.SubmitChanges();
            return true;
        }
        catch
        {
            return false;
        }
    }
}
```

```
    }  
    return false;  
}  
}
```

❖ TaiKhoanController (sau khi sửa)

- ~ Đây là code của phần "nhận lệnh" sau khi đã thuê "người quản lý":

```
// TaiKhoanController.cs (Phiên bản mới, tốt hơn)
```

```
using System.Web.Mvc;
```

```
using WebsiteMovie_DAN.Facades;
```

```
public class TaiKhoanController : Controller
```

```
{
```

```
    private readonly ITaiKhoanFacade _taiKhoanFacade; // Có "người quản lý"
```

```
// Khi tạo TaiKhoanController, phải cung cấp cho nó một "người quản lý"
```

```
    public TaiKhoanController(ITaiKhoanFacade taiKhoanFacade)
```

```
{
```

```
    _taiKhoanFacade = taiKhoanFacade;
```

```
}
```

```
// ... (Các phần khác: xem danh sách, xóa)
```

```
[HttpGet]
```

```
public ActionResult ThemTK()
```

```
{
```

```
    return View(new TaiKhoanViewModel());
```

```
}
```

```
[HttpPost]
```

```
public ActionResult ThemTK(TaiKhoanViewModel viewModel)
```

```
{
```

```
    if (ModelState.IsValid) // Kiểm tra thông tin hợp lệ
```

```
{
```

```
    // "Sai việc" cho người quản lý
```

```
    if (_taiKhoanFacade.ThemTaiKhoan(viewModel))
```

```
{
```

```
        return RedirectToAction("DSTaiKhoan"); // Thêm thành công
```

```
}
```

```
    else
```

```
{
```

```
        ViewBag.ErrorMessage = "Có lỗi xảy ra khi thêm tài khoản."; // Thêm thất bại
```

```
}
```

```
    }

    return View(viewModel);

}

[HttpGet]

public ActionResult SuaTK(string tendn)

{

    var tk = _taiKhoanFacade.LayTaiKhoanTheoTenDangNhap(tendn);

    if (tk == null)

    {

        return HttpNotFound();

    }

    return View(tk);

}
```

```
[HttpPost]

public ActionResult SuaTK(string tendn, TaiKhoanViewModel viewModel)

{

    if (ModelState.IsValid)

    {

        if (_taiKhoanFacade.CapNhatTaiKhoan(viewModel))
```

```
        return RedirectToAction("DSTaiKhoan");

    }

    else

    {

        ViewBag.ErrorMessage = "Có lỗi xảy ra khi cập nhật tài khoản";

    }

}

return View(viewModel);

}

}

// ViewModels/TaiKhoanViewModel.cs

using System.ComponentModel.DataAnnotations;

namespace WebsiteMovie_DAN.ViewModels

{

    public class TaiKhoanViewModel

    {

        [Required(ErrorMessage = "Tên đăng nhập không được để trống")]

        public string TenDN { get; set; }

    }

}
```

```
[Required(ErrorMessage = "Mật khẩu không được để trống")]
```

```
[DataType(DataType.Password)]
```

```
public string MatKhau { get; set; }
```

```
[Required(ErrorMessage = "Email không được để trống")]
```

```
[EmailAddress(ErrorMessage = "Email không hợp lệ")]
```

```
public string Email { get; set; }
```

```
public bool? Quyen { get; set; }
```

```
}
```

```
}
```

~ **Thay đổi:**

- TaiKhoanController không còn "tự làm" mọi việc nữa, mà "sai việc" cho _taiKhoanFacade.
- TaiKhoanController chỉ cần kiểm tra xem thông tin người dùng nhập vào có hợp lệ không (ModelState.IsValid), còn lại giao cho "người quản lý".
- Code trong TaiKhoanController trở nên *ngắn gọn, dễ hiểu* hơn rất nhiều.

6.1.2. Chức năng đăng nhập

- ❖ Code cũ của phần Đăng nhập, đặc biệt là phần "nhận lệnh" (Controller) có tên là DangKyDangNhapController, có một số chỗ chưa tốt:
 - ~ Mã hiện tại của bạn trong DangNhapDangKyController thực hiện quá nhiều việc trong một controller. Nó bao gồm việc:
 - Kiểm tra dữ liệu người dùng (như tên đăng nhập, mật khẩu).
 - Thực hiện các thao tác trực tiếp với cơ sở dữ liệu (kiểm tra trùng tên đăng nhập, lưu tài khoản mới, cập nhật tài khoản).
 - Quản lý logic đăng nhập và đăng ký người dùng.
- Những công việc này làm controller trở nên phức tạp và khó kiểm soát. Để cải thiện, chúng ta có thể áp dụng Facade Pattern để tách biệt các nhiệm vụ.
- ❖ Giải pháp: Dùng "Đăng nhập" (Facade)
 - ~ Để giải quyết các vấn đề trên chúng ta có giải pháp sau:
 - Tạo một **Facade** riêng cho chức năng **Đăng Nhập**.
 - **Controller** chỉ cần giao nhiệm vụ cho **Facade**, thay vì trực tiếp xử lý tất cả các logic như kiểm tra thông tin đăng nhập, kiểm tra mật khẩu và xử lý session.
 - **Facade** sẽ nhận yêu cầu từ controller, thực hiện các thao tác cần thiết và trả về kết quả.
- ❖ Đây là code đầy đủ của phần "nhận lệnh" (DangKyDangNhapController) *trước khi sử dụng mẫu Facade* cho chức năng Đăng Nhập :

// Đây là code cũ, chưa tốt

```
using System; using System.Collections.Generic; using System.Linq; using System.Web;
using System.Web.Mvc; using WebsiteMovie_DAN.Models; using Facebook; using
System.Configuration; using WebsiteMovie_DAN.Common; using System.Security.Policy;

namespace WebsiteMovie_DAN.Controllers { public class DangNhapDangKyController :
Controller { WebmovieDataContext data = new WebmovieDataContext(); // GET:
```

```
DangNhapDangKy private Uri RedirectUri { get { var uriBuilder = new
UriBuilder(Request.Url); uriBuilder.Query = null; uriBuilder.Fragment = null;
uriBuilder.Path = Url.Action("FacebookCallback"); return uriBuilder.Uri; } } public
ActionResult DangNhap() { var tl = data.TheLoais.ToList(); var nam = data.Nams.ToList();
var quocgia = data.QuocGias.ToList(); ViewData["QuocGia"] = quocgia;
ViewData["TheLoai"] = tl; ViewData["Nam"] = nam; return View(); } [HttpPost] public
ActionResult DangNhap(FormCollection collection) { var tl = data.TheLoais.ToList(); var
nam = data.Nams.ToList(); ViewData["TheLoai"] = tl; ViewData["Nam"] = nam; var
quocgia = data.QuocGias.ToList(); ViewData["QuocGia"] = quocgia;

var tendn = collection["TenDN"];
var mk = collection["MatKhau"];
if (String.IsNullOrEmpty(tendn))
    ViewData["Loi"] = "Bạn phải nhập tên đăng nhập";
else if (String.IsNullOrEmpty(mk))
{
    ViewData["Loi1"] = "Bạn phải nhập mật khẩu";
}
else
{
    TaiKhoan tk = data.TaiKhoans.SingleOrDefault(n =>
n.TenDN.Equals(tendn) && n.MatKhau.Equals(SHA_Hash.SHA1(mk)));
    if (tk != null && tk.MatKhau == SHA_Hash.SHA1(mk))
{
        if (tk.Quyen == true)//Admin
{
            @Session["ten"] = tk.TenDN;
            @Session["TK"] = tk.TenDN;
            @Session["quyen"] = 1;
        }
    }
}
```

```
        ViewBag.ThongBao = "Đăng nhập thành công admin";
        return RedirectToAction("Home", "Admin/HomeAdmin/");
    }

    if (tk.Quyen == false || tk.Quyen == null)
    {
        @Session["quyen"] = null;
        @Session["TK"] = tk.TenDN;
        @Session["ten"] = tk.TenDN;
        ViewBag.ThongBao = "Đăng nhập thành công";
        return RedirectToAction("Index", "Home");
    }
}

else
{
    ViewData["Loi2"] = "Tên đăng nhập hoặc mật khẩu không đúng";
}

}

return View();
}

[HttpGet]
public ActionResult DangKy()
{
    var tl = data.TheLoais.ToList();
    var nam = data.Nams.ToList();
    var quocgia = data.QuocGias.ToList();
    ViewData["QuocGia"] = quocgia;
```

```

 ViewData["TheLoai"] = tl;
 ViewData["Nam"] = nam;
 return View();
}

[HttpPost]
public ActionResult DangKy(TaiKhoan tk, FormCollection coll)
{
    var tl = data.TheLoais.ToList();
    var nam = data.Nams.ToList();
    var quocgia = data.QuocGias.ToList();
    ViewData["QuocGia"] = quocgia;
    ViewData["TheLoai"] = tl;
    ViewData["Nam"] = nam;
    var tendn = coll["TenDN"];
    var mk = coll["MatKhau"];
    var mknhaplai = coll["MatKhauNhapLai"];
    var email = coll["Email"];
    //var taikhoan = from t in data.TaiKhoans where t.TenDN.Equals(tendn)
    select t.TenDN;
    var taikhoan = data.TaiKhoans.ToList();
    int kt = 0;
    foreach (var item in taikhoan)
    {
        if (item.TenDN == tendn)
            kt = 1;
    }
    if (String.IsNullOrEmpty(tendn))

```

```
 ViewData["Loi"] = "Chưa điền tên đăng nhập";
else if (String.IsNullOrEmpty(mk))
    ViewData["Loi1"] = "Mật khẩu chưa được điền";
else if (kt == 1)
{
    ViewData["Loi2"] = "Đã có tài khoản này";
}
else if (mk != mknhaplai)
{
    ViewData["Loi12"] = "mật khẩu nhập lại không đúng";
}
else if (String.IsNullOrEmpty(email))
    ViewData["Loi123"] = "Chưa điền thông tin email";
else
{
    tk.TenDN = tendn;
    SHA_Hash Hash = new SHA_Hash();
    tk.MatKhau = SHA_Hash.SHA1(mk);
    tk.Email = email;
    tk.Quyen = false;
    data.TaiKhoans.InsertOnSubmit(tk);
    data.SubmitChanges();
    return RedirectToAction("/DangNhap");
}
if (email == tk.Email)
{
    ViewData["Loi1234"] = "thông tin email bị trùng";
```

```
        }

        return View();
    }

    public ActionResult DangXuat()
    {

        @Session["ten"] = null;
        @Session["quyen"] = null;
        @Session["TK"] = null;

        return Redirect("/");
    }

    public ActionResult DangNhapFaceBook()
    {
        var fb = new FacebookClient();

        var loginUrl = fb.GetLoginUrl(new
        {
            client_id = ConfigurationManager.AppSettings["FbAppId"],
            client_secret = ConfigurationManager.AppSettings["FbAppSecret"],
            redirect_uri = RedirectUri.AbsoluteUri,
            response_type = "code",
            scope = "email",
        });

        return Redirect(loginUrl.AbsoluteUri);
    }

    public ActionResult FacebookCallback(string code, TaiKhoan tk,
FormCollection coll)
{

```

```
var fb = new FacebookClient();
dynamic result = fb.Post("oauth/access_token", new
{
    client_id = ConfigurationManager.AppSettings["FbAppId"],
    client_secret = ConfigurationManager.AppSettings["FbAppSecret"],
    redirect_uri = RedirectUri.AbsoluteUri,
    code = code
});
```

```
var accessToken = result.access_token;
if (!string.IsNullOrEmpty(accessToken))
{
    fb.AccessToken = accessToken;

    dynamic me =
fb.Get("me?fields=first_name,middle_name,last_name,id,email");
    string email = me.email;
    string userName = me.email;
    string firstname = me.first_name;
    string middlename = me.middle_name;
    string lastname = me.last_name;
    string aaaaaaaaaa = firstname + "@321dSDFdgb";
    var taikhoan = data.TaiKhoans.ToList();
    int kt = 0;
    foreach (var item in taikhoan)
    {
        if (item.TenDN == aaaaaaaaaa)
```

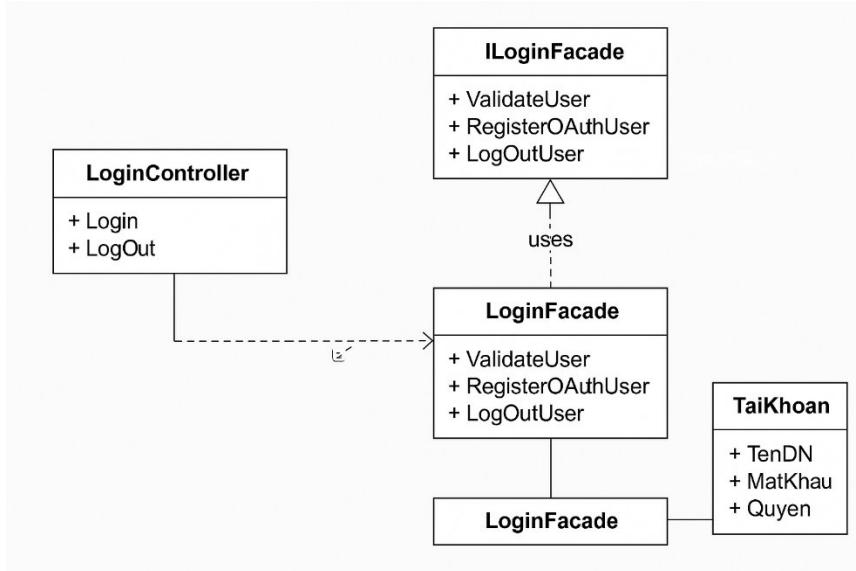
```
        kt = 1;

    }

    if (kt == 1)
    {
        @Session["ten"] = firstname;
        @Session["TK"] = aaaaaaaaa;
        @Session["quyen"] = null;
        data.TaiKhoans.InsertOnSubmit(tk);
    }
    else
    {
        tk.TenDN = firstname + "@321dSDFdgb";
        tk.MatKhau = "12312ABC@312jidqjv";
        data.SubmitChanges();
        @Session["TK"] = firstname;
        @Session["quyen"] = null;
    }
}

return RedirectToAction("Index", "Home");
}
```

❖ Sơ đồ:



Hình 6-2 Sơ đồ đăng nhập

❖ "Bản hợp đồng" (ILoginFacade)

// ILoginFacade.cs

using WebsiteMovie_DAN.Models.Entities; // Đảm bảo sử dụng đúng lớp TaiKhoan từ Entities

namespace WebsiteMovie_DAN.Models.Facade

{

public interface ILoginFacade

{

bool DangNhap(string tenDN, string matKhau, out TaiKhoan taiKhoan); // Sử dụng TaiKhoan từ Entities

}

}

// LoginFacade.cs

using System;

using System.Data.SqlClient;

using BCrypt.Net; // Thêm không gian tên để sử dụng BCrypt

namespace WebsiteMovie_DAN.Models.Facade

{

public class LoginFacade : ILoginFacade

```
{  
    public bool DangNhap(string tenDN, string matKhau, out TaiKhoan taiKhoan)  
    {  
        taiKhoan = null;  
  
        using (var db = new Database()) // Sử dụng class Database  
        {  
            SqlConnection conn = db.GetConnection();  
  
            string query = "SELECT TenDN, MatKhau, Quyen, Email FROM TaiKhoan  
WHERE TenDN = @TenDN";  
            using (SqlCommand cmd = new SqlCommand(query, conn))  
            {  
                cmd.Parameters.AddWithValue("@TenDN", tenDN);  
  
                using (SqlDataReader reader = cmd.ExecuteReader())  
                {  
                    if (reader.Read())  
                    {  
                        string matKhauLuu = reader["MatKhau"].ToString();  
  
                        // Sử dụng BCrypt để kiểm tra mật khẩu đã băm  
                        if (BCrypt.Net.BCrypt.Verify(matKhau, matKhauLuu)) // Kiểm tra mật  
                        khẩu với BCrypt  
                        {  
                            taiKhoan = new TaiKhoan  
                            {  
                                TenDN = reader["TenDN"].ToString(),  
                                MatKhau = matKhauLuu,  
                                Quyen = Convert.ToBoolean(reader["Quyen"]),  
                                Email = reader["Email"].ToString()  
                            };  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

```

        return true;
    }
}
}
}
}
return false;
}
}
}
}
}

```

❖ LoginController (sau khi xóa bỏ DangKyDangNhapController và tạo class mới là LoginController)

```

using System.Web.Mvc;
using WebsiteMovie_DAN.Models;
using WebsiteMovie_DAN.Models.Facade;
using WebsiteMovie_DAN.Models.Repository;

namespace WebsiteMovie_DAN.Controllers
{
    public class LoginController : Controller
    {
        private readonly LoginFacade facade = new LoginFacade(); //không cần dependency
        injection
        private readonly ITaiKhoanRepository _taiKhoanRepository = new
            TaiKhoanRepository();
        // GET: Login
        public ActionResult DangNhap()
        {
            return View();
        }
    }
}

```

```
[HttpPost]
```

```
public ActionResult DangNhap(string tenDN, string matKhau)
```

```
{
```

```
    if (facade.DangNhap(tenDN, matKhau, out TaiKhoan taiKhoan))
```

```
{
```

```
    Session["ten"] = taiKhoan.TenDN;
```

```
    Session["TK"] = (taiKhoan.Quyen ?? false) ? "Admin" : "User";
```

```
    return RedirectToAction("Index", "Home");
```

```
}
```

```
ViewBag.ThongBao = "Tên đăng nhập hoặc mật khẩu không đúng!";
```

```
return View();
```

```
}
```

```
public ActionResult DangXuat()
```

```
{
```

```
    Session.Clear();
```

```
    return RedirectToAction("DangNhap");
```

```
}
```

```
[HttpGet]
```

```
public ActionResult DangKy()
```

```
{
```

```
    return View();
```

```
}
```

```
[HttpPost]
```

```
public ActionResult DangKy(string tenDN, string matKhau, string email)
```

```
{
```

```
    var taiKhoan = new TaiKhoan
```

```
{
```

```
        TenDN = tenDN,
```

```
        MatKhau = matKhau, // Chưa mã hóa tại đây, mã hóa sẽ được thực hiện trong
        Repository
        Quyen = false, // Mặc định user thường
        Email = email
    };

    if (_taiKhoanRepository.DangKy(taiKhoan))
    {
        ViewBag.ThongBao = "Đăng ký thành công!";
        return RedirectToAction("DangNhap");
    }
    else
    {
        ViewBag.ThongBao = "Tên đăng nhập đã tồn tại!";
    }

    return View();
}
}
```

6.1.3. Chức năng Tin tức

Đầu tiên ta có code trước khi áp dụng Facade:

```
public ActionResult TinTuc()  
  
{  
  
    var DSPhimBo = data.DSPhimBos.OrderByDescending(x => x.LuotXem).Take(3).ToList();  
  
    ViewData["TopPhim"] = DSPhimBo;  
  
  
  
  
    var tl = data.TheLoais.ToList();  
  
    var nam = data.Nams.ToList();  
  
    var quocgia = data.QuocGias.ToList();  
  
  
  
  
    ViewData["TheLoai"] = tl;  
  
    ViewData["Nam"] = nam;  
  
    ViewData["QuocGia"] = quocgia;  
  
  
  
  
    var tt = data.tintucphims.ToList();  
  
    return View(tt);  
  
}  
  
  
  
  
public ActionResult ChiTietTinTuc(int id)
```

```
{  
  
    var DSPhimBo = data.DSPhimBos.OrderByDescending(x => x.LuotXem).Take(3).ToList();  
  
    ViewData["TopPhim"] = DSPhimBo;  
  
  
  
    var tl = data.TheLoais.ToList();  
  
    var nam = data.Nams.ToList();  
  
    var quocgia = data.QuocGias.ToList();  
  
  
  
  
    ViewData["TheLoai"] = tl;  
  
    ViewData["Nam"] = nam;  
  
    ViewData["QuocGia"] = quocgia;  
  
  
  
  
    var tt = data.tintucphims.SingleOrDefault(n => n.idtintuc == id);  
  
    return View(tt);  
  
}
```

Nhược điểm

Controller phụ thuộc vào Database (WebmovieDataContext) → Khi thay đổi database, Controller cũng phải sửa.

Code lặp lại nhiều lần: Đoạn code lấy danh sách **TopPhim**, **ThểLoại**, **Năm**, **QuốcGia** bị

lặp ở cả hai phương thức TinTuc() và ChiTietTinTuc().

Khó mở rộng: Nếu cần thêm tính năng lọc dữ liệu hoặc phân trang, phải sửa lại nhiều nơi trong Controller.

Code Mới (Sau Khi Áp Dụng Facade + Repository):

Đặc điểm:

Tách biệt truy vấn dữ liệu ra khỏi Controller → Sử dụng Facade và Repository.

Code Controller đơn giản hơn, chỉ cần gọi _tinTucFacade và _newRepo.

Dễ mở rộng, dễ thay đổi database hoặc source dữ liệu mà không ảnh hưởng Controller

Code mới trong TinTucPhimController:

```
public class TinTucPhimController : Controller
{
    private readonly ITinTucFacade _tinTucFacade;
    private readonly IRepository<tintucphim> _newRepo;

    public TinTucPhimController(ITinTucFacade tinTucFacade, IRepository<tintucphim>
newRepo)
    {
        _tinTucFacade = tinTucFacade;
        _newRepo = newRepo;
    }

    public ActionResult TinTuc()
    {
        var newList = _newRepo.GetAll().ToList();
        Create ViewData();
    }
}
```

```

        return View(newList);
    }

public ActionResult ChiTietTinTuc(int id)
{
    var data = _newRepo.Find(x => x.idtintuc == id).FirstOrDefault();
    if (data == null)
    {
        return HttpNotFound();
    }
    ViewData["TopPhim"] = data.TopPhimBo;
    ViewData["TheLoai"] = data.TheLoais;
    ViewData["Nam"] = data.Nams;
    ViewData["QuocGia"] = data.QuocGias;
}
}

private void Create ViewData()
{
    var data = _tinTucFacade.GetTinTucData();
    ViewData["TopPhim"] = data.TopPhimBo;
    ViewData["TheLoai"] = data.TheLoais;
    ViewData["Nam"] = data.Nams;
    ViewData["QuocGia"] = data.QuocGias;
}
}

```

Code trong TinTucFacade:

```

public class TinTucFacade : ITinTucFacade
{
    private readonly WebmovieDataContext _data;

```

```

public TinTucFacade(WebmovieDataContext data)
{
    _data = data;
}

public TinTucData GetTinTucData()
{
    return new TinTucData
    {
        TopPhimBo = _data.DSPhimBos.OrderByDescending(x => x.LuotXem).Take(3).ToList(),
        TheLoais = _data.TheLoais.ToList(),
        Nams = _data.Nams.ToList(),
        QuocGias = _data.QuocGias.ToList()
    };
}

```

Code trong GenericRepository<T>

```

public class GenericRepository<T> : IRepository<T> where T : class
{
    private readonly WebmovieDataContext _context;
    private readonly DbSet<T> _dbSet;

    public GenericRepository(WebmovieDataContext context)
    {
        _context = context;
        _dbSet = context.Set<T>();
    }
}

```

```
public IEnumerable<T> GetAll()
{
    return _dbSet.ToList();
}

public IEnumerable<T> Find(Func<T, bool> predicate)
{
    return _dbSet.Where(predicate).ToList();
}

public void Insert(T entity)
{
    _dbSet.Add(entity);
    _context.SaveChanges();
}

public void Update(T entity)
{
    _context.Entry(entity).State = EntityState.Modified;
    _context.SaveChanges();
}

public void Delete(T entity)
{
    _dbSet.Remove(entity);
    _context.SaveChanges();
}
```

Lợi Ích Sau Khi Áp Dụng

1. Controller trở nên nhẹ nhàng

Không chứa truy vấn SQL.

Chỉ gọi Facade và Repository.

2. Dễ bảo trì & mở rộng

Nếu cần thêm tính năng mới, chỉ cần chỉnh sửa trong Facade hoặc Repository.

Không cần sửa lại Controller.

3. Tái sử dụng code tốt hơn

GenericRepository<T> có thể dùng cho **tất cả các bảng** trong database.

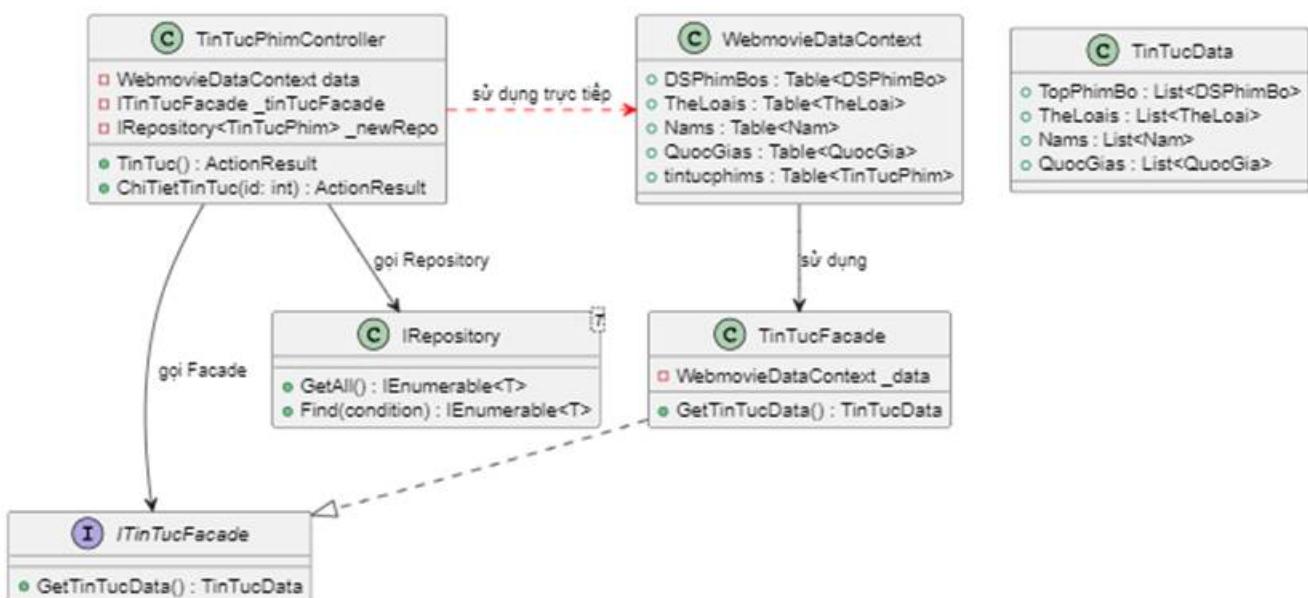
TinTucFacade có thể tái sử dụng ở nhiều Controller khác.

4. Hỗ trợ Dependency Injection

Controller không phụ thuộc vào WebmovieDataContext.

Có thể dễ dàng **Mock dữ liệu để test** mà không cần kết nối database.

SƠ ĐỒ:



Hình 6-3 Sơ đồ tin tức

Trước khi áp dụng Facade (dòng màu đỏ, nét đứt):

TinTucPhimController **trực tiếp** sử dụng WebmovieDataContext, gây phụ thuộc chặt chẽ.

Sau khi áp dụng Facade (phần còn lại):

TinTucPhimController **không còn dùng WebmovieDataContext trực tiếp** nữa.

Thay vào đó, nó **gọi ITinTucFacade** để lấy dữ liệu từ TinTucFacade.

TinTucFacade sử dụng WebmovieDataContext để lấy dữ liệu, giúp giảm phụ thuộc.

Lợi ích của sơ đồ gộp

Hiển thị cả trước và sau khi cải tiến trên cùng một sơ đồ.

Dễ so sánh sự thay đổi trong kiến trúc.

Giúp hiểu rõ cách Facade giúp giảm sự phụ thuộc vào WebmovieDataContext.

6.2 Áp dụng mẫu Strategy

6.2.1. Chức năng quản lý thể loại và năm phát hành

❖ Trong dự án website xem phim, phần code quản lý thể loại (thêm, sửa, xóa, hiển thị danh sách thể loại) nếu làm theo cách thông thường sẽ dễ bị:

- ~ **Lặp lại code:** Các thao tác thêm, sửa, xóa, ... khá giống nhau giữa các danh mục (thể loại, quốc gia, năm sản xuất...).
- ~ **Khó bảo trì:** Khi cần thay đổi, phải sửa nhiều chỗ.
- ~ **Khó mở rộng:** Thêm danh mục mới (ví dụ: diễn viên) sẽ tốn nhiều công.

❖ Giải pháp: Strategy Pattern

~ Để giải quyết các vấn đề trên, chúng ta sẽ áp dụng *Strategy Pattern*. Hãy hình dung:

- Chúng ta có *nhiều cách* để quản lý các danh mục khác nhau (thể loại, quốc gia,...).
- Mỗi "cách" đó được gọi là một *strategy*.
- Sẽ có một "người quản lý" chọn "cách" phù hợp.

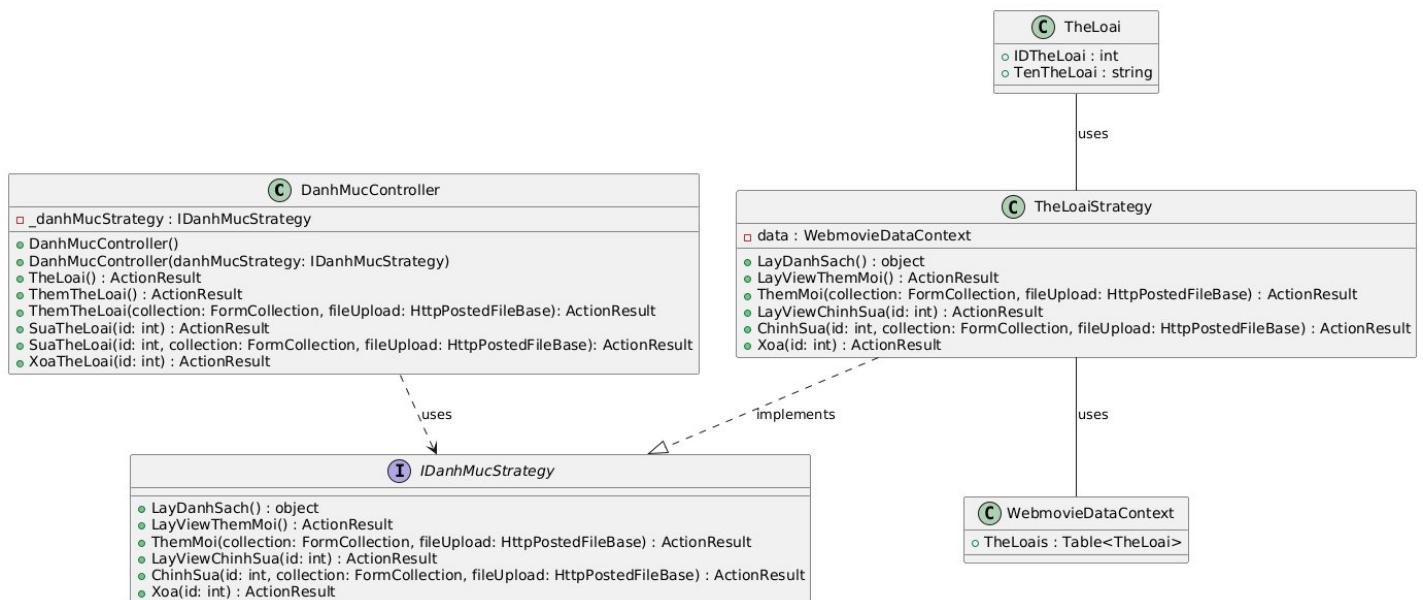
~ Nói một cách chuyên môn hơn, *Strategy Pattern* là một mẫu thiết kế giúp định nghĩa một tập hợp các thuật toán, đóng gói chúng lại, và làm cho chúng có thể hoán đổi cho nhau.

~ Cách áp dụng là chúng ta sẽ tạo ra:

- **Một "bản hợp đồng" (**IDanhMucStrategy**):** Quy định những việc mà một "người quản lý danh mục" *phải làm được*.
- **Các "người quản lý" cụ thể (các **Strategy**):** Hiện tại, chúng ta có *TheLoaiStrategy* chuyên quản lý thể loại. Sau này có thể thêm

QuocGiaStrategy, NamSanXuatStrategy,...

❖ Sơ đồ:



Hình 6-4 Sơ đồ quản lý thể loại

❖ "Bản Hợp Đồng" - IDanhMucStrategy

```

namespace WebsiteMovie_DAN.Areas.Admin.Strategy_Pattern {
    public interface IDanhMucStrategy {
        object LayDanhSach(); // Lấy danh sách ActionResult
        LayViewThemMoi(); // Chuẩn bị giao diện thêm mới ActionResult
        ThemMoi(FormCollection collection, HttpPostedFileBase fileUpload); // Xử lý
        // thêm ActionResult
        LayViewChinhSua(int id); // Chuẩn bị giao diện chỉnh sửa
        ActionResult ChinhSua(int id, FormCollection collection, HttpPostedFileBase
        fileUpload); // Xử lý chỉnh sửa ActionResult
        Xoa(int id); // Xử lý xóa
    }
}
  
```

~ **Giải thích:** `IDanhMucStrategy` định nghĩa các hàm (phương thức) mà *bất kỳ*

"người quản lý danh mục" nào cũng phải có:

- LayDanhSach(): Trả về danh sách (thể loại, quốc gia,...).
- LayViewThemMoi(), ThemMoi(): Liên quan đến việc thêm mới.
- LayViewChinhSua(), ChinhSua(): Liên quan đến việc chỉnh sửa.
- Xoa(): Liên quan đến việc xóa.

6.2.1. Chức năng quản lý thể loại và năm phát hành

❖ "Người Quản Lý Thể Loại" - TheLoaiStrategy

```
using System; using System.Collections.Generic; using System.Linq; using
System.Web; using System.Web.Mvc; using WebsiteMovie_DAN.Models;

namespace WebsiteMovie_DAN.Areas.Admin.Strategy_Pattern { public class
TheLoaiStrategy : IDanhMucStrategy { private WebmovieDataContext data =
new WebmovieDataContext();

public object LayDanhSach()
{
    return data.TheLoais.ToList();
}

public ActionResult LayViewThemMoi()
{
    return new ViewResult { ViewName = "ThemTheLoai" };
}

public ActionResult ThemMoi(FormCollection collection, HttpPostedFileBase
fileUpload)
{
    var tenTheLoai = collection["TenTheLoai"];
```

```
var danhSachTheLoai = data.TheLoais.ToList();

if (string.IsNullOrEmpty(tenTheLoai))
{
    return new ViewResult { ViewName = "ThemTheLoai", ViewData = new
ViewDataDictionary { { "Loi1", "Tên thể loại không được để trống" } } };
}

foreach (var theLoai in danhSachTheLoai)
{
    if (string.Compare(theLoai.TenTheLoai, tenTheLoai, true) == 0)
    {
        return new ViewResult { ViewName = "ThemTheLoai", ViewData = new
ViewDataDictionary { { "Loi2", "Đã có thể loại này" } } };
    }
}

TheLoai tl = new TheLoai { TenTheLoai = tenTheLoai };
data.TheLoais.InsertOnSubmit(tl);
data.SubmitChanges();
return new RedirectToRouteResult(new
System.Web.Routing.RouteValueDictionary { { "controller", "DanhMuc" },
{ "action", "TheLoai" } });
}

public ActionResult LayViewChinhSua(int id)
{
    var theLoai = data.TheLoais.FirstOrDefault(n => n.IDTheLoai == id);
```

```
if (theLoai == null)
{
    return new HttpNotFoundResult(); // Trả về trang 404
}

return new ViewResult { ViewName = "SuaTheLoai", ViewData = new
ViewDataDictionary(theLoai) }; // Truyền model trực tiếp
}

public ActionResult ChinhSua(int id, FormCollection collection,
HttpPostedFileBase fileUpload)
{
    var theLoaiHienTai = data.TheLoais.FirstOrDefault(n => n.IDTheLoai == id);
    var tenTheLoaiMoi = collection["TenTheLoai"];
    var danhSachTheLoai = data.TheLoais.ToList();

    foreach (var theLoai in danhSachTheLoai)
    {
        if (string.Compare(theLoai.TenTheLoai, tenTheLoaiMoi, true) == 0 &&
theLoai.IDTheLoai != id)
        {
            return new ViewResult { ViewName = "SuaTheLoai", ViewData = new
ViewDataDictionary(theLoaiHienTai) { { "Loi2", "Tên thể loại đã tồn tại" } } };
        }
    }

    if (string.IsNullOrEmpty(tenTheLoaiMoi))
    {
        return new ViewResult { ViewName = "SuaTheLoai", ViewData = new
```

```
 ViewDataDictionary(theLoaiHienTai) { { "Loi1", "Tên thể loại không được để  
trống" } } };  
}
```

```
theLoaiHienTai.TenTheLoai = tenTheLoaiMoi;  
data.SubmitChanges();  
return new RedirectToRouteResult(new  
System.Web.Routing.RouteValueDictionary { { "controller", "DanhMuc" },  
{ "action", "TheLoai" } });  
}
```

```
public ActionResult Xoa(int id)  
{  
    TheLoai theLoai = data.TheLoais.SingleOrDefault(n => n.IDTheLoai == id);  
    if (theLoai == null)  
    {  
        return new HttpNotFoundResult();  
    }  
    data.TheLoais.DeleteOnSubmit(theLoai);  
    data.SubmitChanges();  
    return new RedirectToRouteResult(new  
System.Web.Routing.RouteValueDictionary { { "controller", "DanhMuc" },  
{ "action", "TheLoai" } });  
}  
}  
  
}
```

~

Giải thích:

- TheLoaiStrategy làm theo "bản hợp đồng" IDanhMucStrategy.
- Nó chứa code để thực hiện các thao tác thêm, sửa, xóa, ... riêng cho thể loại.
- Nó sử dụng WebmovieDataContext để kết nối và làm việc với cơ sở dữ liệu.
- Chi tiết từng hàm:
 - LayDanhSach(): Lấy danh sách thể loại từ cơ sở dữ liệu và chuyển đổi sang dạng List.
 - LayViewThemMoi(): Trả về View "ThemTheLoai" để hiển thị form thêm mới.
 - ThemMoi(FormCollection collection, HttpPostedFileBase fileUpload):
 - Lấy tên thể loại từ FormCollection.
 - Kiểm tra tên thể loại có bị trống không.
 - Kiểm tra tên thể loại đã tồn tại chưa.
 - Nếu hợp lệ, tạo đối tượng TheLoai mới và thêm vào cơ sở dữ liệu.
 - Chuyển hướng về trang danh sách thể loại.
 - LayViewChinhSua(int id):
 - Tìm thể loại theo id.
 - Nếu tìm thấy, trả về View "SuaTheLoai" với dữ liệu thể loại.
 - Nếu không tìm thấy, trả về HttpNotFoundResult.
 - ChinhSua(int id, FormCollection collection, HttpPostedFileBase fileUpload):
 - Tìm thể loại cần sửa.

- Lấy tên thẻ loại mới từ form.
- Kiểm tra tính hợp lệ của tên thẻ loại mới (không trùng, không trùng lặp).
- Cập nhật tên thẻ loại trong cơ sở dữ liệu.
- Chuyển hướng người dùng về trang danh sách.

❖ DanhMucController

```
namespace WebsiteMovie_DAN.Areas.Admin.Controllers
{
    public class DanhMucController : Controller
    {
        private WebmovieDataContext data = new WebmovieDataContext();
        private IDanhMucStrategy _strategy;
        public DanhMucController()
        {
            _strategy = new TheLoaiStrategy(); // Mặc định là thẻ loại
        }
        public DanhMucController(IDanhMucStrategy strategy)
```

```
{  
    _strategy = strategy;  
}  
  
// ===== THÊ LOẠI =====  
  
public ActionResult TheLoai()  
{  
    _strategy = new TheLoaiStrategy();  
  
    var danhSach = _strategy.LayDanhSach();  
  
    return View(danhSach);  
}  
  
[HttpGet]  
  
public ActionResult ThemTheLoai()  
{  
    _strategy = new TheLoaiStrategy();  
  
    return _strategy.LayViewThemMoi();  
}  
  
[HttpPost]  
  
public ActionResult ThemTheLoai/FormCollection collection, HttpPostedFileBase  
fileUpload = null)
```

```
{  
    _strategy = new TheLoaiStrategy();  
  
    return _strategy.ThemMoi(collection, fileUpload);  
}
```

[HttpGet]

```
public ActionResult SuaTheLoai(int id)
```

```
{  
    _strategy = new TheLoaiStrategy();  
  
    return _strategy.LayViewChinhSua(id);  
}
```

[HttpPost]

```
public ActionResult SuaTheLoai(int id, FormCollection collection, HttpPostedFileBase  
fileUpload = null)
```

```
{  
    _strategy = new TheLoaiStrategy();  
  
    return _strategy.ChinhSua(id, collection, fileUpload);  
}
```

```
public ActionResult XoaTheLoai(int id)
```

```
{
```

```
_strategy = new TheLoaiStrategy();

return _strategy.Xoa(id);

}
```

```
// ===== NĂM PHÁT HÀNH =====
```

```
public ActionResult NamPhatHanh()

{

    _strategy = new NamPhatHanhStrategy();

    var danhSach = _strategy.LayDanhSach();

    return View(danhSach);

}
```

```
[HttpGet]
```

```
public ActionResult ThemNam()

{

    _strategy = new NamPhatHanhStrategy();

    return _strategy.LayViewThemMoi();

}
```

```
[HttpPost]
```

```
public ActionResult ThemNam(FormCollection collection, HttpPostedFileBase
fileUpload = null)
```

```
{  
    _strategy = new NamPhatHanhStrategy();  
  
    return _strategy.ThemMoi(collection, fileUpload);  
}
```

[HttpGet]

```
public ActionResult SuaNam(int id)
```

```
{  
    _strategy = new NamPhatHanhStrategy();  
  
    return _strategy.LayViewChinhSua(id);  
}
```

[HttpPost]

```
public ActionResult SuaNam(int id, FormCollection collection, HttpPostedFileBase  
fileUpload = null)
```

```
{  
    _strategy = new NamPhatHanhStrategy();  
  
    return _strategy.ChinhSua(id, collection, fileUpload);  
}
```

```
public ActionResult XoaNam(int id)
```

```
{
```

```
_strategy = new NamPhatHanhStrategy();  
  
return _strategy.Xoa(id);  
  
}
```

Giải thích:

- DanhMucController không còn trực tiếp làm các công việc thêm, sửa, xóa danh mục nữa.
- Nó có một biến `_strategy` để giữ "bản hợp đồng" IDanhMucStrategy.
- Constructor mặc định khởi tạo `_strategy` với TheLoaiStrategy.
- Constructor thứ hai cho phép inject một strategy khác.
- Các action trong controller gọi các phương thức của `_strategy` để thực hiện các thao tác quản lý danh mục.
- Điểm quan trọng: Mỗi action đều tạo mới một instance của strategy. Điều này không hiệu quả vì nó tạo ra các đối tượng không cần thiết và có thể ảnh hưởng đến hiệu suất.
- Khi người dùng muốn xem danh sách danh mục, DanhMucController chỉ việc "gọi" hàm LayDanhSach() của `_strategy`. Nó không cần biết chi tiết bên trong hàm đó làm gì. Tương tự với các hàm thêm, sửa, xóa.
- Constructor (Cách 2) cho phép truyền vào một "người quản lý danh mục" khác (ví dụ, NamPhatHanhStrategy) nếu sau này bạn cần.
- Các action được chia thành hai nhóm: quản lý thẻ loại và quản lý năm phát hành.
- Mỗi nhóm có các action tương ứng cho việc hiển thị danh sách, thêm, sửa và xóa.
- Các action sử dụng ViewResult để trả về view và RedirectToRouteResult để chuyển hướng sau khi hoàn thành các thao tác.
- Các action HttpGet trả về view, và các action HttpPost xử lý dữ liệu được gửi từ form.

6.2.2. Chức năng tìm kiếm theo danh mục

```
//code theo mẫu Strategy

private readonly TimKiemContext timKiemContext = new TimKiemContext();
public ActionResult TimKiem(string type, int id)

{

    // Cấu hình Strategy theo loại tìm kiếm

    switch (type)

    {

        case "TheLoai":

            timKiemContext.SetStrategy(new TimKiemTheoTheLoai());

            break;

        case "Nam":

            timKiemContext.SetStrategy(new TimKiemTheoNam());

            break;

        case "QuocGia":

            timKiemContext.SetStrategy(new TimKiemTheoQuocGia());

            break;

        default:

            return HttpNotFound();

    }

    // Lấy dữ liệu phim theo chiến lược đã chọn
}
```

```

var phim = timKiemContext.TimKiem(data, id);

// Lấy dữ liệu hiển thị

ViewData["TopPhim"] = data.DSPhimBos.OrderByDescending(x =>
    x.LuotXem).Take(3).ToList();

ViewData["TheLoai"] = data.TheLoais.ToList();

ViewData["Nam"] = data.Nams.ToList();

ViewData["QuocGia"] = data.QuocGias.ToList();

return View(phim);
}

```

6.2.2. Chức năng Tìm kiếm

- **Khởi tạo đối tượng Strategy Context:**

Một đối tượng TimKiemContext được khởi tạo sẵn (có thể coi như bộ điều phối), chịu trách nhiệm gọi chiến lược tìm kiếm phù hợp.

- **Xác định loại tìm kiếm dựa vào type:**

Dựa trên tham số type nhận được từ request, chương trình kiểm tra người dùng muốn tìm theo tiêu chí nào. Có 3 lựa chọn:

- a. "TheLoai": Tìm theo thể loại.
- b. "Nam": Tìm theo năm phát hành.
- c. "QuocGia": Tìm theo quốc gia. Với mỗi trường hợp, hệ thống gán chiến lược tương ứng bằng cách gọi timKiemContext.SetStrategy(...).

- **Trả về lỗi nếu loại không hợp lệ:**

Nếu type không thuộc một trong các giá trị đã định nghĩa, chương trình trả về lỗi

HttpNotFound().

- **Thực hiện tìm kiếm với chiến lược đã chọn:**

Sau khi gán chiến lược, chương trình gọi timKiemContext.TimKiem(data, id) để thực hiện tìm kiếm phim, trong đó id là giá trị cụ thể cần tìm (ví dụ ID thể loại, ID quốc gia,...).

- **Chuẩn bị dữ liệu phụ trợ cho View:**

Các danh sách như top 3 phim bộ có lượt xem cao nhất, danh sách thể loại, năm và quốc gia cũng được truyền qua ViewData để hỗ trợ hiển thị giao diện.

- **Trả về View kèm kết quả tìm kiếm:**

Cuối cùng, kết quả tìm kiếm phim sẽ được truyền vào View(phim) để hiển thị trên trang web.

6.2.3. Chức năng tìm kiếm ở trang chủ

```
public interface ITimKiemStrategy
```

```
{
```

```
    List<object> TimKiem(string keyword);
```

```
}
```

```
public class TimKiemPhimBo : ITimKiemStrategy
```

```
{
```

```
    private WebmovieDataContext data;
```

```
    public TimKiemPhimBo(WebmovieDataContext db)
```

```
{
```

```
        data = db;
```

```
}

public List<object> TimKiem(string keyword)

{
    return data.DSPhimBos.Where(m =>
        m.TenPhim.Contains(keyword)).Cast<object>().ToList();
}

public class TimKiemPhimLe : ITimKiemStrategy

{
    private WebmovieDataContext data;

    public TimKiemPhimLe(WebmovieDataContext db)
    {
        data = db;
    }

    public List<object> TimKiem(string keyword)
    {
        return data.DSPhimLes.Where(m =>
            m.TenPhim.Contains(keyword)).Cast<object>().ToList();
    }
}
```

```
}
```

```
public class TimKiemContext
```

```
{
```

```
    private ITimKiemStrategy _strategy;
```

```
    public void SetStrategy(ITimKiemStrategy strategy)
```

```
{
```

```
    _strategy = strategy;
```

```
}
```

```
    public List<object> ThucHienTimKiem(string keyword)
```

```
{
```

```
    return _strategy?.TimKiem(keyword) ?? new List<object>();
```

```
}
```

```
}
```

```
    public ActionResult TimKiem/FormCollection c)
```

```
{
```

```
        var DSPhimBo = data.DSPhimBos.OrderByDescending(x =>
            x.LuotXem).Take(3).ToList();
```

```
        ViewData["TopPhim"] = DSPhimBo;
```

```
        ViewData["QuocGia"] = data.QuocGias.ToList();
```

```
        ViewData["TheLoai"] = data.TheLoais.ToList();
```

```
 ViewData["Nam"] = data.Nams.ToList();

var tim = c["tim"];
if (string.IsNullOrEmpty(tim))
{
    return View(new List<object>());
}

// Strategy Pattern

var timKiemContext = new TimKiemContext();
timKiemContext.SetStrategy(new TimKiemPhimBo(data));
var phimBo = timKiemContext.ThucHienTimKiem(tim);
timKiemContext.SetStrategy(new TimKiemPhimLe(data));
var phimLe = timKiemContext.ThucHienTimKiem(tim);

var ketQua = phimBo.Concat(phimLe).ToList();
return View(ketQua);
}

public ActionResult LuotXemTinTuc(int id)
```

```
{  
    tintucphim phim = data.tintucphims.SingleOrDefault(n => n.idtintuc == id);  
  
    phim.luotxem += 1;  
  
    UpdateModel(phim);  
  
    data.SubmitChanges();  
  
    return RedirectToAction("ChiTietTinTuc", "TinTucPhim", new { id = phim.idtintuc });  
}
```

Interface ITimKiemStrategy:

Đây là giao diện định nghĩa một phương thức TimKiem(string keyword) mà tất cả các chiến lược tìm kiếm phải triển khai.

Phương thức trả về một danh sách List<object> để dễ kết hợp nhiều loại dữ liệu khác nhau (phim bộ, phim lẻ,...).

Lớp TimKiemPhimBo và TimKiemPhimLe:

Cả hai đều triển khai ITimKiemStrategy.

TimKiemPhimBo tìm kiếm các bộ phim trong bảng DSPhimBos theo từ khóa tên phim.

TimKiemPhimLe tìm kiếm các phim lẻ trong bảng DSPhimLes.

Kết quả được ép kiểu về object để thông nhất định dạng dữ liệu trả về.

Lớp TimKiemContext:

Là lớp trung gian (context) chứa một đối tượng chiến lược tìm kiếm _strategy.

Có phương thức SetStrategy(...) để thay đổi chiến lược tại runtime.

Phương thức ThucHienTimKiem(...) gọi thực hiện tìm kiếm theo chiến lược đang được gán.

Action TimKiem(FormCollection c):

Lấy danh sách top 3 phim bộ có lượt xem cao nhất và các dữ liệu phụ như thể loại, quốc gia, năm để hiển thị.

Nhận dữ liệu từ form tìm kiếm qua c["tim"].

Nếu ô tìm kiếm trống thì trả về view rỗng.

Gọi TimKiemContext và lần lượt sử dụng hai chiến lược: tìm kiếm phim bộ và tìm kiếm phim lẻ.

Kết quả hai loại phim được gộp lại bằng Concat và truyền vào View.

Action LuotXemTinTuc(int id):

Dùng để tăng lượt xem của một bản tin phim (tintucphim) khi người dùng nhấn vào tin đó.

Tìm tin tức theo id, tăng thuộc tính luotxem lên 1, cập nhật vào database.

Sau đó chuyển hướng đến trang chi tiết tin tức tương ứng.

6.3 Áp dụng mẫu Repository

6.3.1 Chức năng Đăng ký

❖ Code cũ của phần Đăng Ký, đặc biệt là phần "nhận lệnh" (Controller) có tên là DangKyDangNhapController, có một số chỗ chưa tốt:

~ Mã hiện tại của bạn trong DangNhapDangKyController thực hiện quá nhiều việc

trong một controller. Nó bao gồm việc:

- Kiểm tra dữ liệu người dùng (như tên đăng nhập, mật khẩu).
- Thực hiện các thao tác trực tiếp với cơ sở dữ liệu (kiểm tra trùng tên đăng nhập, lưu tài khoản mới, cập nhật tài khoản).
- Quản lý logic đăng nhập và đăng ký người dùng.

Những công việc này làm controller trở nên phức tạp và khó kiểm soát. Để cải thiện, chúng ta có thể áp dụng Repository để tách biệt các nhiệm vụ.

Giải pháp: Dùng "Đăng Ký" (Repository)

Mã hiện tại của bạn trong DangNhapDangKyController thực hiện quá nhiều việc trong một controller. Nó bao gồm việc:

- **Tạo một lớp Repository** để xử lý tất cả các thao tác liên quan đến việc lưu trữ, kiểm tra tên đăng nhập trùng lặp và mã hóa mật khẩu.
- **Controller** sẽ giao các nhiệm vụ liên quan đến cơ sở dữ liệu cho **Repository**, giúp controller chỉ cần nhận yêu cầu từ người dùng và xử lý kết quả.

Các lớp cần thiết

TaiKhoanRepository: Chứa các phương thức truy cập cơ sở dữ liệu liên quan đến tài khoản, như kiểm tra sự tồn tại của tên đăng nhập, lưu tài khoản mới và các thao tác liên quan.

ITaiKhoanRepository: Interface định nghĩa các phương thức của TaiKhoanRepository.

LoginController: Controller sẽ giao nhiệm vụ cho TaiKhoanRepository và chỉ tập trung vào việc nhận yêu cầu từ người dùng, xử lý kết quả và trả về View.

// Đây là code cũ, chưa tốt

```
using System; using System.Collections.Generic; using System.Linq; using
System.Web; using System.Web.Mvc; using WebsiteMovie_DAN.Models; using
Facebook; using System.Configuration; using WebsiteMovie_DAN.Common; using
System.Security.Policy;
```

```

namespace WebsiteMovie_DAN.Controllers { public class
DangNhapDangKyController : Controller { WebmovieDataContext data = new
WebmovieDataContext(); // GET: DangNhapDangKy private Uri RedirectUri { get
{ var uriBuilder = new UriBuilder(Request.Url); uriBuilder.Query = null;
uriBuilder.Fragment = null; uriBuilder.Path = Url.Action("FacebookCallback");
return uriBuilder.Uri; } } public ActionResult DangNhap() { var tl =
data.TheLoais.ToList(); var nam = data.Nams.ToList(); var quocgia =
data.QuocGias.ToList(); ViewData["QuocGia"] = quocgia; ViewData["TheLoai"] =
tl; ViewData["Nam"] = nam; return View(); } [HttpPost] public ActionResult
DangNhap(FormCollection collection) { var tl = data.TheLoais.ToList(); var nam =
data.Nams.ToList(); ViewData["TheLoai"] = tl; ViewData["Nam"] = nam; var
quocgia = data.QuocGias.ToList(); ViewData["QuocGia"] = quocgia;

var tendn = collection["TenDN"];
var mk = collection["MatKhau"];
if (String.IsNullOrEmpty(tendn))
    ViewData["Loi"] = "Bạn phải nhập tên đăng nhập";
else if (String.IsNullOrEmpty(mk))
{
    ViewData["Loi1"] = "Bạn phải nhập mật khẩu";
}
else
{
    TaiKhoan tk = data.TaiKhoans.SingleOrDefault(n =>
n.TenDN.Equals(tendn) &&
n.MatKhau.Equals(SHA_Hash.SHA1(mk)));
    if (tk != null && tk.MatKhau == SHA_Hash.SHA1(mk))
{
        if (tk.Quyen == true)//Admin
{

```

```
        @Session["ten"] = tk.TenDN;
        @Session["TK"] = tk.TenDN;
        @Session["quyen"] = 1;
        ViewBag.ThongBao = "Đăng nhập thành công admin";
        return RedirectToAction("Home",
        "Admin/HomeAdmin/");
    }

    if (tk.Quyen == false || tk.Quyen == null)
    {
        @Session["quyen"] = null;
        @Session["TK"] = tk.TenDN;
        @Session["ten"] = tk.TenDN;
        ViewBag.ThongBao = "Đăng nhập thành công";
        return RedirectToAction("Index", "Home");
    }
}

else
{
    ViewData["Loi2"] = "Tên đăng nhập hoặc mật khẩu
không đúng";
}

return View();
}

[HttpGet]
public ActionResult DangKy()
```

```
{  
    var tl = data.TheLoais.ToList();  
    var nam = data.Nams.ToList();  
    var quocgia = data.QuocGias.ToList();  
    ViewData["QuocGia"] = quocgia;  
    ViewData["TheLoai"] = tl;  
    ViewData["Nam"] = nam;  
    return View();  
}  
[HttpPost]  
public ActionResult DangKy(TaiKhoan tk, FormCollection coll)  
{  
    var tl = data.TheLoais.ToList();  
    var nam = data.Nams.ToList();  
    var quocgia = data.QuocGias.ToList();  
    ViewData["QuocGia"] = quocgia;  
    ViewData["TheLoai"] = tl;  
    ViewData["Nam"] = nam;  
    var tendn = coll["TenDN"];  
    var mk = coll["MatKhau"];  
    var mknhaplai = coll["MatKhauNhapLai"];  
    var email = coll["Email"];  
    //var taikhoan = from t in data.TaiKhoans where  
    t.TenDN.Equals(tendn) select t.TenDN;  
    var taikhoan = data.TaiKhoans.ToList();  
    int kt = 0;  
    foreach (var item in taikhoan)
```

```
{  
    if (item.TenDN == tendn)  
        kt = 1;  
    }  
    if (String.IsNullOrEmpty(tendn))  
        ViewData["Loi"] = "Chưa điền tên đăng nhập";  
    else if (String.IsNullOrEmpty(mk))  
        ViewData["Loi1"] = "Mật khẩu chưa được điền";  
    else if (kt == 1)  
    {  
        ViewData["Loi2"] = "Đã có tài khoản này";  
    }  
    else if (mk != mknhaplai)  
    {  
        ViewData["Loi12"] = "mật khẩu nhập lại không đúng";  
    }  
    else if (String.IsNullOrEmpty(email))  
        ViewData["Loi123"] = "Chưa điền thông tin email";  
    else  
    {  
        tk.TenDN = tendn;  
        SHA_Hash Hash = new SHA_Hash();  
        tk.MatKhau = SHA_Hash.SHA1(mk);  
        tk.Email = email;  
        tk.Quyen = false;  
        data.TaiKhoans.InsertOnSubmit(tk);  
        data.SubmitChanges();
```

```
        return RedirectToAction("/DangNhap");

    }

    if (email == tk.Email)

    {

        ViewData["Loi1234"] = "thông tin email bị trùng";

    }

    return View();

}

public ActionResult DangXuat()

{

    @Session["ten"] = null;

    @Session["quyen"] = null;

    @Session["TK"] = null;

    return Redirect("/");

}

public ActionResult DangNhapFaceBook()

{

    var fb = new FacebookClient();

    var loginUrl = fb.GetLoginUrl(new

    {

        client_id = ConfigurationManager.AppSettings["FbAppId"],

        client_secret =



ConfigurationManager.AppSettings["FbAppSecret"],

        redirect_uri = RedirectUri.AbsoluteUri,

        response_type = "code",

        scope = "email",
```

```
});

        return Redirect(loginUrl.AbsoluteUri);
    }

    public ActionResult FacebookCallback(string code, TaiKhoan tk,
FormCollection coll)
{
    var fb = new FacebookClient();
    dynamic result = fb.Post("oauth/access_token", new
    {
        client_id = ConfigurationManager.AppSettings["FbAppId"],
        client_secret =
ConfigurationManager.AppSettings["FbAppSecret"],
        redirect_uri = RedirectUri.AbsoluteUri,
        code = code
    });

    var accessToken = result.access_token;
    if (!string.IsNullOrEmpty(accessToken))
    {
        fb.AccessToken = accessToken;

        dynamic me =
fb.Get("me?fields=first_name,middle_name,last_name,id,email");
        string email = me.email;
        string userName = me.email;
        string firstname = me.first_name;
```

```
        string middlename = me.middle_name;
        string lastname = me.last_name;
        string aaaaaaaaaa = firstname + "@321dSDFdgb";
        var taikhoan = data.TaiKhoans.ToList();
        int kt = 0;
        foreach (var item in taikhoan)
        {
            if (item.TenDN == aaaaaaaaaa)
                kt = 1;
        }

        if (kt == 1)
        {
            @Session["ten"] = firstname;
            @Session["TK"] = aaaaaaaaaa;
            @Session["quyen"] = null;
            data.TaiKhoans.InsertOnSubmit(tk);
        }
        else
        {
            tk.TenDN = firstname + "@321dSDFdgb";
            tk.MatKhau = "12312ABC@312jidqjv";
            data.SubmitChanges();
            @Session["TK"] = firstname;
            @Session["quyen"] = null;
        }
    }
}
```

```
        return RedirectToAction("Index", "Home");  
    }  
}  
  
}
```

❖ "Bản Hợp Đồng" - ITaiKhoanRepository

```
using System;  
  
using System.Collections.Generic;  
  
using System.Linq;  
  
using System.Web;
```

```
namespace WebsiteMovie_DAN.Models.Repository
```

```
{  
  
    public interface ITaiKhoanRepository  
    {  
  
        bool KiemTraTonTai(string tenDN);  
  
        bool DangKy(TaiKhoan taiKhoan);  
  
    }  
  
}
```

❖ "Đăng ký tài khoản" - TaiKhoanRepository

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.Helpers;

namespace WebsiteMovie_DAN.Models.Repository
{
    public class TaiKhoanRepository : ITaiKhoanRepository
    {
        public bool KiemTraTonTai(string tenDN)
        {
            using (var db = new Database())
            {
                SqlConnection conn = db.GetConnection();
                string query = "SELECT COUNT(*) FROM TaiKhoan WHERE TenDN =
@TenDN";

                using (SqlCommand cmd = new SqlCommand(query, conn))
                {
                    cmd.Parameters.AddWithValue("@TenDN", tenDN);
                    int count = (int)cmd.ExecuteScalar();
                    return count > 0;
                }
            }
        }
    }
}
```

```
public bool DangKy(TaiKhoan taiKhoan)
{
    if (KiemTraTonTai(taiKhoan.TenDN))
        return false;

    using (var db = new Database())
    {
        SqlConnection conn = db.GetConnection();
        string query = "INSERT INTO TaiKhoan (TenDN, MatKhau, Quyen, Email)
VALUES (@TenDN, @MatKhau, @Quyen, @Email)";

        using (SqlCommand cmd = new SqlCommand(query, conn))
        {
            string hashedPassword =
BCrypt.Net.BCrypt.HashPassword(taiKhoan.MatKhau);

            cmd.Parameters.AddWithValue("@TenDN", taiKhoan.TenDN);
            cmd.Parameters.AddWithValue("@MatKhau", hashedPassword);
            cmd.Parameters.AddWithValue("@Quyen", taiKhoan.Quyen);
            cmd.Parameters.AddWithValue("@Email", taiKhoan.Email);

            return cmd.ExecuteNonQuery() > 0;
        }
    }
}
```

❖ LoginController (sau khi xóa bỏ DangKyDangNhapController và tạo class mới là

```
    LoginController)

using System.Web.Mvc;

using WebsiteMovie_DAN.Models;

using WebsiteMovie_DAN.Models.Facade;

using WebsiteMovie_DAN.Models.Repository;

namespace WebsiteMovie_DAN.Controllers

{

    public class LoginController : Controller

    {

        private readonly LoginFacade facade = new LoginFacade(); //không cần dependency
        injection

        private readonly ITaiKhoanRepository _taiKhoanRepository = new
        TaiKhoanRepository();

        // GET: Login

        public ActionResult DangNhap()

        {

            return View();

        }

        [HttpPost]

        public ActionResult DangNhap(string tenDN, string matKhau)
```

```
{  
    if (facade.DangNhap(tenDN, matKhau, out TaiKhoan taiKhoan))  
    {  
        Session["ten"] = taiKhoan.TenDN;  
        Session["TK"] = (taiKhoan.Quyen ?? false) ? "Admin" : "User";  
        return RedirectToAction("Index", "Home");  
    }  
}
```

```
ViewBag.ThongBao = "Tên đăng nhập hoặc mật khẩu không đúng!";  
return View();  
}
```

```
public ActionResult DangXuat()  
{  
    Session.Clear();  
    return RedirectToAction("DangNhap");  
}  
}
```

```
[HttpGet]  
public ActionResult DangKy()  
{  
    return View();  
}
```

```
}
```

```
[HttpPost]
```

```
public ActionResult DangKy(string tenDN, string matKhau, string email)
```

```
{
```

```
    var taiKhoan = new TaiKhoan
```

```
{
```

```
    TenDN = tenDN,
```

```
    MatKhau = matKhau, // Chưa mã hóa tại đây, mã hóa sẽ được thực hiện trong  
    Repository
```

```
    Quyen = false, // Mặc định user thường
```

```
    Email = email
```

```
};
```

```
if (_taiKhoanRepository.DangKy(taiKhoan))
```

```
{
```

```
    ViewBag.ThongBao = "Đăng ký thành công!";
```

```
    return RedirectToAction("DangNhap");
```

```
}
```

```
else
```

```
{
```

```
    ViewBag.ThongBao = "Tên đăng nhập đã tồn tại!";
```

```
}
```

```
        return View();  
    }  
}  
}
```

6.3.2 Chức năng Lượt thích

Code trước khi áp dụng Respository:

```
public ActionResult YeuThichPhimBo(string tendn)  
{  
    var DSPhimBo = data.DSPhimBos.OrderByDescending(x =>  
        x.LuotThich).Take(3).ToList();  
  
    ViewData["TopPhim"] = DSPhimBo;  
  
    var tl = data.TheLoais.ToList();  
  
    var nam = data.Nams.ToList();  
  
    var quocgia = data.QuocGias.ToList();  
  
    ViewData["TheLoai"] = tl;  
  
    ViewData["Nam"] = nam;  
  
    ViewData["QuocGia"] = quocgia;  
  
    var yeuthich = data.LuotThichPhimBos.Where(m => m.TenDN.Equals(tendn)).ToList();  
}
```

```
        return View(yeuthich);
    }

// Thêm yêu thích phim bộ

public ActionResult ThemYeuThichPhimBo(string tendn, int idphim, int id)

{
    var quocgia = data.QuocGias.ToList();
    var tl = data.TheLoais.ToList();
    var nam = data.Nams.ToList();

    ViewData["TheLoai"] = tl;
    ViewData["Nam"] = nam;
    ViewData["QuocGia"] = quocgia;

    LuotThichPhimBo yeuthich = new LuotThichPhimBo();
    var dsphim = data.LuotThichPhimBos.Where(m => m.TenDN == tendn).ToList();
    DSPhimBo a = data.DSPhimBos.SingleOrDefault(n => n.ID == id);

    foreach (var item in dsphim)
    {
        if (item.IDPhim == idphim)
        {
            a.LuotThich += 1;
        }
    }
}
```

```
        UpdateModel(a);

        data.SubmitChanges();

        return RedirectToAction("ChiTietPhim", "ChiTietPhim", new { id = a.ID });

    }

}

yeuthich.TenDN = tendn;
yeuthich.IDPhim = idphim;
data.LuotThichPhimBos.InsertOnSubmit(yeuthich);
data.SubmitChanges();

a.LuotThich += 1;

UpdateModel(a);

data.SubmitChanges();

return RedirectToAction("ChiTietPhim", "ChiTietPhim", new { id = a.ID });

}

// Xóa yêu thích phim bộ

public ActionResult XoaYeuThichPhimBo(string tendn, int idphim)
{
    LuotThichPhimBo pb = data.LuotThichPhimBos.SingleOrDefault(n => n.TenDN ==
tendn && n.IDPhim == idphim);
```

```

if (pb == null)
{
    Response.SubStatusCode = 404;
    return null;
}

data.LuotThichPhimBos.DeleteOnSubmit(pb);
data.SubmitChanges();

return RedirectToAction("YeuThichPhimBo", new { tendn = pb.TenDN });
}

```

Nhược điểm của code cũ (chưa dùng Repository Pattern)

Vi phạm Nguyên Tắc Đóng/Mở (OCP) → Khó mở rộng, phải sửa nhiều code nếu thay đổi cách truy xuất dữ liệu.

Controller quá tải với logic dữ liệu → Gây rối, khó bảo trì.

Phụ thuộc chặt vào WebmovieDataContext → Khó thay đổi database, khó test.

Khó kiểm thử (Unit Testing) → Không thể mock dữ liệu để kiểm tra logic.

Lặp lại code truy vấn dữ liệu → Code dư thừa, khó bảo trì.

Áp dụng Repository Pattern vào **LuotThich (Like feature)** giúp:

Tách biệt logic truy vấn dữ liệu – Controller chỉ gọi phương thức Repository thay vì truy vấn trực tiếp vào database.

Giảm lặp lại mã nguồn – Gom nhóm các thao tác like/unlike vào Repository, tránh lặp lại logic.

Dễ bảo trì và mở rộng – Dễ dàng thay đổi cách lưu dữ liệu (SQL, MongoDB, cache) mà không cần sửa Controller.

Tăng hiệu suất – Có thể kết hợp **Proxy Pattern** để cache dữ liệu, giảm truy vấn database.

Để kiểm thử (Unit Test) – Cho phép mock Repository để kiểm thử mà không cần kết nối database.

Đầu tiên ta tạo class **Interface Repository Chung**

Mục đích:

Tạo một interface IRepository<T> dùng chung để thao tác với database.

Giúp tái sử dụng cho nhiều bảng dữ liệu khác nhau.

Interface này chứa các phương thức chung để thao tác với dữ liệu như: Lấy dữ liệu (GetAll()), Thêm (Insert()), Sửa (Update()), Xóa (Delete()), Lưu (Save()).

T là một lớp dữ liệu bất kỳ, giúp tái sử dụng linh hoạt.

```
public interface IRepository<T> where T : class  
{  
    IEnumerable<T> GetAll();  
  
    T GetById(object id);  
  
    IEnumerable<T> Find(Func<T, bool> predicate);  
  
    void Insert(T entity);  
  
    void Update(T entity);  
  
    void Delete(object id);  
  
    void Save();  
}
```

Tiếp theo, Tạo Generic Repository (GenericRepository<T>):

Mục đích:

Triển khai IRepository<T> để thực hiện các thao tác với database.

Tránh lặp code khi làm việc với nhiều bảng dữ liệu.

GenericRepository<T> kế thừa từ IRepository<T>, giúp thao tác dữ liệu dễ dàng.

context: Đối tượng kết nối database (WebmovieDataContext).

table: Bảng dữ liệu lấy từ database.

Các phương thức cơ bản:

GetAll() → Lấy toàn bộ dữ liệu từ bảng.

GetById(id) → Tìm dữ liệu theo ID.

Find(predicate) → Lọc dữ liệu theo điều kiện.

Insert(entity), Delete(id), Save() → Thêm, xóa, lưu dữ liệu.

```
public class GenericRepository<T> : IRepository<T> where T : class
{
    protected readonly WebmovieDataContext _context;
    protected readonly Table<T> _table;

    public GenericRepository(WebmovieDataContext context)
    {
        _context = context;
        _table = _context.GetTable<T>();
    }

    public IEnumerable<T> GetAll() => _table.ToList();

    public T GetById(object id) => _table.SingleOrDefault(e =>
```

```

GetEntityId(e).Equals(id));

public IEnumerable<T> Find(Func<T, bool> predicate) =>
    _table.Where(predicate).ToList();

public void Insert(T entity) => _table.InsertOnSubmit(entity);

public void Delete(object id)

{
    var entity = GetById(id);

    if (entity != null) _table.DeleteOnSubmit(entity);

}

public void Save() => _context.SubmitChanges();
}

```

Tiếp tục Tạo Repository Cho "Lượt Thích":

Mục đích:

Xây dựng repository riêng cho tính năng **Lượt Thích (Like)** của phim.

Kế thừa **GenericRepository<LuotThichPhimBo>** để sử dụng lại code.

Thêm các chức năng đặc biệt như **thêm/xóa lượt thích, lấy danh sách phim được like nhiều nhất.**

GetTopLikedPhimBo(int count) → Lấy danh sách phim được like nhiều nhất.

GetUserLikes(string tenDN) → Lấy danh sách phim mà user đã like.

AddLike(string tenDN, int idPhim) → Thêm lượt thích.

RemoveLike(string tenDN, int idPhim) → Xóa lượt thích.

```
public interface ILuotYeuThichRepository : IRepository<LuotThichPhimBo>
{
    IEnumerable<DSPhimBo> GetTopLikedPhimBo(int count);
    IEnumerable<LuotThichPhimBo> GetUserLikes(string tenDN);
    void AddLike(string tenDN, int idPhim);
    void RemoveLike(string tenDN, int idPhim);
}

public class LuotYeuThichRepository : GenericRepository<LuotThichPhimBo>, ILuotYeuThichRepository
{
    private readonly IRepository<DSPhimBo> _phimBoRepo;

    public LuotYeuThichRepository(WebmovieDataContext context, IRepository<DSPhimBo> phimBoRepo)
        : base(context)
    {
        _phimBoRepo = phimBoRepo;
    }

    public void AddLike(string tenDN, int idPhim)
    {
        var existingLike = Find(m => m.TenDN == tenDN && m.IDPhim ==
    }
```

```

idPhim).FirstOrDefault();

    if (existingLike == null)

    {

        Insert(new LuotThichPhimBo { TenDN = tenDN, IDPhim = idPhim });

    }

}

var phimBo = _phimBoRepo.Find(x => x.ID == idPhim).FirstOrDefault();

if (phimBo != null) phimBo.LuotThich += 1;

_phimBoRepo.Save();

}

public void RemoveLike(string tenDN, int idPhim)

{

    var like = Find(m => m.TenDN == tenDN && m.IDPhim ==

idPhim).FirstOrDefault();

    if (like != null) _table.DeleteOnSubmit(like);

    _context.SubmitChanges();

}

}

```

Thêm like: Kiểm tra nếu user chưa like phim thì **thêm vào bảng LuotThichPhimBo**.

Tăng số lượt thích của phim.

Xóa like: Nếu user đã like phim, xóa khỏi bảng LuotThichPhimBo.

Cuối cùng Cập Nhật Controller (LuotThichController)

Mục đích:

Sử dụng LuotYeuThichRepository để gọi các phương thức like/unlike.
Tránh gọi database trực tiếp trong Controller.

```
public class LuotThichController : Controller  
{  
    private readonly ILuotYeuThichRepository _repository;  
  
    public LuotThichController(ILuotYeuThichRepository repository)  
    {  
        _repository = repository;  
    }  
  
    public ActionResult YeuThichPhimBo(string tendn)  
    {  
        var yeuthich = _repository.GetUserLikes(tendn).ToList();  
        return View(yeuthich);  
    }  
}
```

```

public ActionResult ThemYeuThichPhimBo(string tendn, int idphim, int id)

{
    _repository.AddLike(tendn, idphim);

    return RedirectToAction("ChiTietPhim", "ChiTietPhim", new { id });
}

public ActionResult XoaYeuThichPhimBo(string tendn, int idphim)

{
    _repository.RemoveLike(tendn, idphim);

    return RedirectToAction("YeuThichPhimBo", new { tendn });

}

```

YeuThichPhimBo() → Hiển thị danh sách phim mà user đã like.

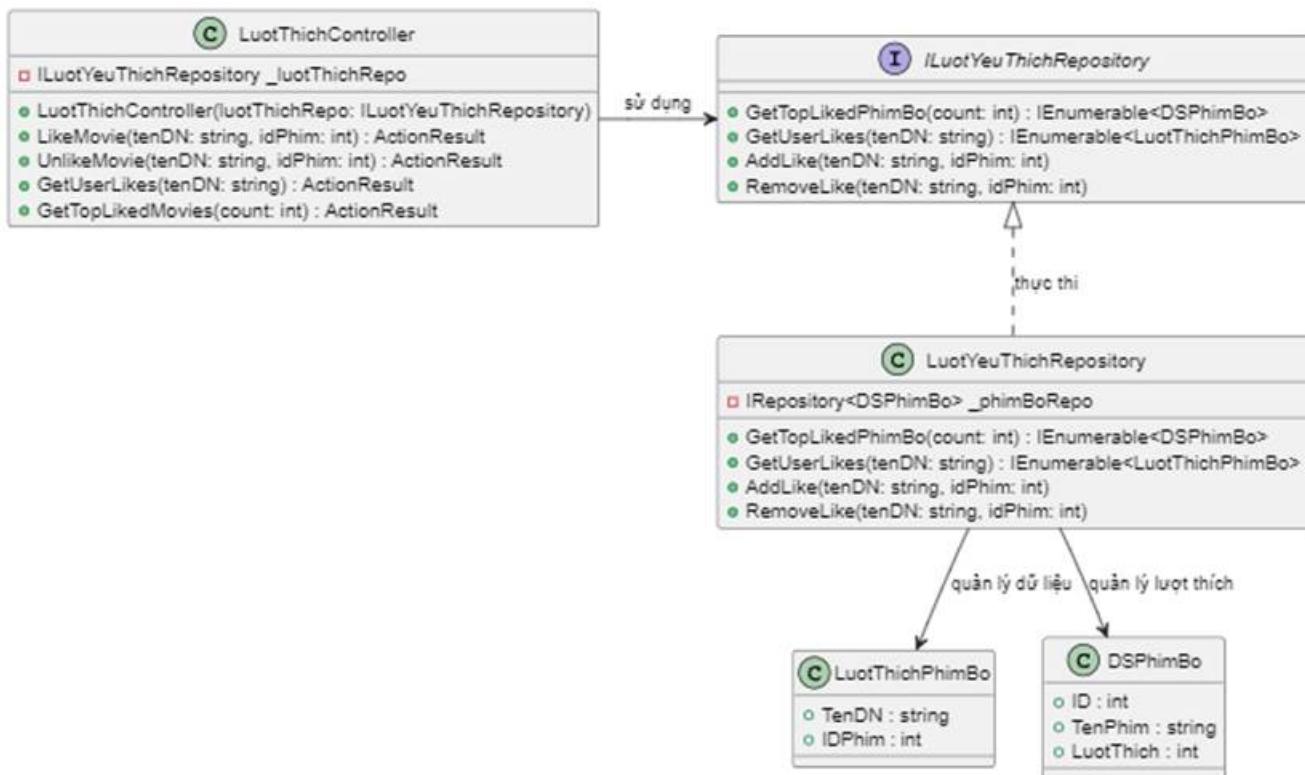
ThemYeuThichPhimBo() → Gọi `_repository.AddLike()` để thêm like.

XoaYeuThichPhimBo() → Gọi `_repository.RemoveLike()` để bỏ like.

Lợi ích:

- ✓ **Code gọn gàng hơn** (giảm trùng lặp).
- ✓ **Dễ bảo trì** (thay đổi 1 nơi, ảnh hưởng toàn hệ thống).
- ✓ **Dễ mở rộng** (thêm tính năng mà không phá vỡ hệ thống cũ).
- ✓ **Hỗ trợ viết Unit Test dễ dàng.**

Sơ Đồ:



Hình 6-5 Sơ đồ lượt yêu thích

Giải thích sơ đồ gộp chung

Trước khi áp dụng Facade (dòng màu đỏ, nét đứt):

TinTucPhimController trực tiếp sử dụng WebmovieDataContext, gây phụ thuộc chặt chẽ.

Sau khi áp dụng Facade (phần còn lại):

TinTucPhimController không còn dùng WebmovieDataContext trực tiếp nữa.

Thay vào đó, nó gọi ITinTucFacade để lấy dữ liệu từ TinTucFacade.

TinTucFacade sử dụng WebmovieDataContext để lấy dữ liệu, giúp giảm phụ thuộc.

Lợi ích của sơ đồ gộp

Hiển thị cả trước và sau khi cải tiến trên cùng một sơ đồ.

Dễ so sánh sự thay đổi trong kiến trúc.

Giúp hiểu rõ cách Facade giúp giảm sự phụ thuộc vào WebmovieDataContext.

6.4 Áp dụng mẫu Proxy:

Đầu tiên ta có code trước khi áp dụng Facade:

```
public ActionResult TinTuc()
{
    var DSPhimBo = data.DSPhimBos.OrderByDescending(x => x.LuotXem).Take(3).ToList();

    ViewData["TopPhim"] = DSPhimBo;

    var tl = data.TheLoais.ToList();
    var nam = data.Nams.ToList();
    var quocgia = data.QuocGias.ToList();

    ViewData["TheLoai"] = tl;
    ViewData["Nam"] = nam;
    ViewData["QuocGia"] = quocgia;

    var tt = data.tintucphims.ToList();

    return View(tt);
}

public ActionResult ChiTietTinTuc(int id)
```

```
{  
  
    var DSPhimBo = data.DSPhimBos.OrderByDescending(x => x.LuotXem).Take(3).ToList();  
  
    ViewData["TopPhim"] = DSPhimBo;  
  
  
  
  
    var tl = data.TheLoais.ToList();  
  
    var nam = data.Nams.ToList();  
  
    var quocgia = data.QuocGias.ToList();  
  
  
  
  
    ViewData["TheLoai"] = tl;  
  
    ViewData["Nam"] = nam;  
  
    ViewData["QuocGia"] = quocgia;  
  
  
  
  
    var tt = data.tintucphims.SingleOrDefault(n => n.idtintuc == id);  
  
    return View(tt);  
  
}
```

Nhược điểm

Controller phụ thuộc vào Database (WebmovieDataContext) → Khi thay đổi database, Controller cũng phải sửa.

Code lặp lại nhiều lần: Đoạn code lấy danh sách **TopPhim**, **ThểLoại**, **Năm**, **QuốcGia** bị

lặp ở cả hai phương thức TinTuc() và ChiTietTinTuc().

Khó mở rộng: Nếu cần thêm tính năng lọc dữ liệu hoặc phân trang, phải sửa lại nhiều nơi trong Controller.

Code Mới (Sau Khi Áp Dụng Facade + Repository):

Đặc điểm:

Tách biệt truy vấn dữ liệu ra khỏi Controller → Sử dụng Facade và Repository.

Code Controller đơn giản hơn, chỉ cần gọi _tinTucFacade và _newRepo.

Dễ mở rộng, dễ thay đổi database hoặc source dữ liệu mà không ảnh hưởng Controller

Code mới trong TinTucPhimController:

```
public class TinTucPhimController : Controller
{
    private readonly ITinTucFacade _tinTucFacade;
    private readonly IRepository<tintucphim> _newRepo;

    public TinTucPhimController(ITinTucFacade tinTucFacade, IRepository<tintucphim>
newRepo)
    {
        _tinTucFacade = tinTucFacade;
        _newRepo = newRepo;
    }

    public ActionResult TinTuc()
    {
        var newList = _newRepo.GetAll().ToList();
        Create ViewData();
        return View(newList);
    }

    public ActionResult ChiTietTinTuc(int id)
    {
        var data = _newRepo.Find(x => x.idtintuc == id).FirstOrDefault();
        if (data == null)
        {
            return HttpNotFound();
        }
        Create ViewData();
        return View(data);
    }

    private void Create ViewData()
    {
```

```

var data = _tinTucFacade.GetTinTucData();
ViewData["TopPhim"] = data.TopPhimBo;
ViewData["TheLoai"] = data.TheLoais;
ViewData["Nam"] = data.Nams;
ViewData["QuocGia"] = data.QuocGias;
}
}

```

Code trong TinTucFacade:

```

public class TinTucFacade : ITinTucFacade
{
    private readonly WebmovieDataContext _data;

    public TinTucFacade(WebmovieDataContext data)
    {
        _data = data;
    }

    public TinTucData GetTinTucData()
    {
        return new TinTucData
        {
            TopPhimBo = _data.DSPhimBos.OrderByDescending(x => x.LuotXem).Take(3).ToList(),
            TheLoais = _data.TheLoais.ToList(),
            Nams = _data.Nams.ToList(),
            QuocGias = _data.QuocGias.ToList()
        };
    }
}

```

Code trong GenericRepository<T>

```
public class GenericRepository<T> : IRepository<T> where T : class
{
    private readonly WebmovieDataContext _context;
    private readonly DbSet<T> _dbSet;

    public GenericRepository(WebmovieDataContext context)
    {
        _context = context;
        _dbSet = context.Set<T>();
    }

    public IEnumerable<T> GetAll()
    {
        return _dbSet.ToList();
    }

    public IEnumerable<T> Find(Func<T, bool> predicate)
    {
        return _dbSet.Where(predicate).ToList();
    }

    public void Insert(T entity)
    {
        _dbSet.Add(entity);
        _context.SaveChanges();
    }

    public void Update(T entity)
    {
        _context.Entry(entity).State = EntityState.Modified;
        _context.SaveChanges();
    }
}
```

```
public void Delete(T entity)
{
    _dbSet.Remove(entity);
    _context.SaveChanges();
}
```

Lợi Ích Sau Khi Áp Dụng

1. Controller trở nên nhẹ nhàng

Không chứa truy vấn SQL.

Chỉ gọi Facade và Repository.

2. Dễ bảo trì & mở rộng

Nếu cần thêm tính năng mới, chỉ cần chỉnh sửa trong Facade hoặc Repository.

Không cần sửa lại Controller.

3. Tái sử dụng code tốt hơn

GenericRepository<T> có thể dùng cho **tất cả các bảng** trong database.

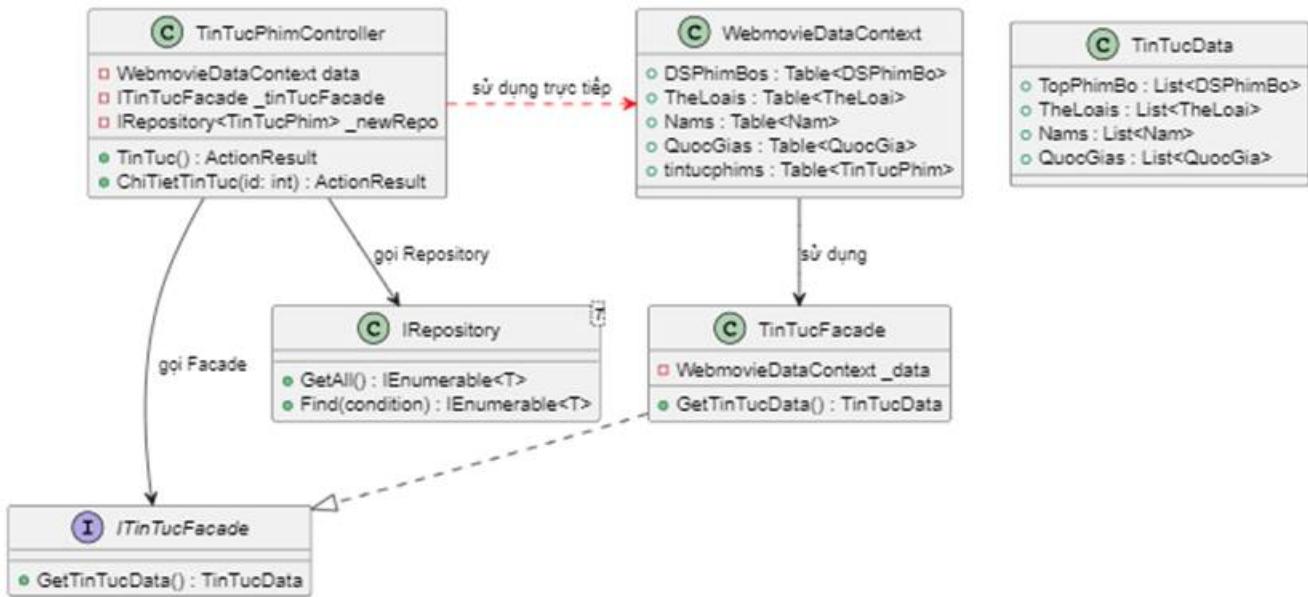
TinTucFacade có thể tái sử dụng ở nhiều Controller khác.

4. Hỗ trợ Dependency Injection

Controller không phụ thuộc vào WebmovieDataContext.

Có thể dễ dàng **Mock dữ liệu để test** mà không cần kết nối database.

SƠ ĐỒ:



Hình 6-6 Sơ đồ lượt thích

Trước khi áp dụng Facade (dòng màu đỏ, nét đứt):

TinTucPhimController **trực tiếp** sử dụng WebmovieDataContext, gây phụ thuộc chặt chẽ.

Sau khi áp dụng Facade (phần còn lại):

TinTucPhimController **không còn dùng WebmovieDataContext trực tiếp** nữa.

Thay vào đó, nó **gọi ITinTucFacade** để lấy dữ liệu từ TinTucFacade.

TinTucFacade sử dụng WebmovieDataContext để lấy dữ liệu, giúp giảm phụ thuộc.

Lợi ích của sơ đồ gộp

Hiển thị cả **trước và sau khi cải tiến** trên cùng một sơ đồ.

Dễ so sánh sự thay đổi trong kiến trúc.

Giúp hiểu rõ cách **Facade giúp giảm sự phụ thuộc** vào WebmovieDataContext.

7. KẾT LUẬN - HƯỚNG PHÁT TRIỂN

7.1. Các chức năng đã hoàn thành

- Phía người mua: hệ thống đáp ứng được các yêu cầu cơ bản như đăng nhập, đăng ký, tra cứu các thông tin cần thiết, mua hàng, thanh toán qua VNPay, xem lại các hóa đơn đã đặt, cập nhật các thông tin của hóa đơn, lưu và sử dụng voucher, thêm sản phẩm vào mục yêu thích.
- Phía người bán: Hệ thống cung cấp các chức năng cơ bản cần thiết như tra cứu / thêm / xóa / sửa các sản phẩm, đơn hàng, tạo voucher, thống kê doanh thu, . Xây dựng một hệ thống đáp ứng đầy đủ các nhu cầu cơ bản của một website bán hàng.
- Phía quản trị: hệ thống cung cấp các chức năng cơ bản cần thiết như tra cứu phê duyệt các sản phẩm, đơn hàng, tạo voucher, thống kê doanh thu, quản lý người dùng, quản lý phân quyền, quản lý shipper, quản lý tin tức, doanh thu... . Xây dựng một hệ thống đáp ứng đầy đủ các nhu cầu cơ bản của một website bán hàng.

7.2. Các chức năng chưa được thực hiện

- Hệ thống dữ liệu còn nhiều thiếu sót, chỉ mới đáp ứng được các nhu cầu cơ bản, chưa thể hiện tính linh động khi bảo trì nâng cấp hệ thống.
- Hệ thống chưa hỗ trợ tính năng đa ngôn ngữ.
- Chưa có chức năng phân tích được thói quen, sở thích của khách hàng để đưa ra các gợi ý cho khách hàng.
- Chức năng thanh toán chưa hỗ trợ đầy đủ cho các kênh thanh toán.

7.3. Các chức năng có thể phát triển

- Hỗ trợ đa ngôn ngữ cho hệ thống.
- Phân tích lịch sử mua hàng đưa ra các ưu đãi phù hợp nhu cầu đến người dùng.
- Thêm các kênh thanh toán cho khách hàng dễ dàng lựa chọn.

TÀI LIỆU THAM KHẢO

<https://refactoring.guru/design-patterns/strategy>

<https://www.geeksforgeeks.org/strategy-pattern-set-1/>

https://www.tutorialspoint.com/design_pattern/strategy_pattern.htm

<https://java-design-patterns.com/patterns/strategy/>