

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. Mục tiêu và nội dung đồ án

- a. Mục tiêu:
- b. Nội dung:

2. Kết quả đạt được

- a.
- b.
- c.

3. Ý thức làm việc của sinh viên

- a.
- b.
- c.

Hà Nội, ngày ... tháng ... năm 2024
Giảng viên hướng dẫn

Vũ Thành Nam

Mở đầu

Lời đầu tiên, em xin gửi lời cảm ơn chân thành và sâu sắc tới thầy – TS. Vũ Thành Nam vì đã tận tình hỗ trợ, giúp đỡ em trong quá trình thực hiện đồ án. Qua thời gian làm việc với thầy trong đồ án lần này, em đã học thêm được nhiều kiến thức hữu ích về Phân tích thiết kế hệ thống thông tin. Những kiến thức và trải nghiệm này là một bài học quý báu để em có thể phát triển bản thân hơn cho các báo cáo lần sau. Em rất mong sẽ có nhiều cơ hội làm việc với thầy hơn trong tương lai.

Bên cạnh đó, em cũng xin gửi lời cảm ơn đến các thầy cô trong Khóa Toán-Tin, Đại học Bách khoa Hà Nội vì đã cung cấp cho em những kiến thức nền tảng đủ vững chắc để hoàn thiện đồ án này. Đồ án là kết quả của nhiều nền tảng kiến thức từ nhiều học phần khác nhau, nhờ sự hướng dẫn và giảng dạy tận tình của các thầy cô, em đã có cơ hội áp dụng những kiến thức này vào đồ án một cách hiệu quả.

Trong quá trình hoàn thiện đồ án, dù đã chuẩn bị khá kỹ lưỡng nhưng chắc chắn sẽ không tránh khỏi những thiếu sót. Em rất mong có thể nhận được nhiều góp ý từ các thầy cô để đồ án có thể hoàn thiện hơn, qua đó em cũng rút ra được kinh nghiệm và phát triển bản thân hơn.

Em xin chân thành cảm ơn!

Hà Nội, ngày 21 tháng 6 năm 2024
Sinh viên thực hiện đồ án

Cao Bảo Nguyên

Mục lục

Mở đầu	3
Danh sách hình vẽ.....	7
Bảng thuật ngữ.....	10
Phần 1. Khảo sát	11
1.1 Ứng dụng CNTT vào quản lý rạp chiếu phim.....	11
1.2 Khảo sát hệ thống hiện tại của CGV	11
1.2.1 Phạm vi khảo sát	11
1.2.2 Cơ cấu tổ chức	12
1.2.3 Các quy trình nghiệp vụ chính	12
1.3 Đề xuất hệ thống	13
Phần 2. Xác định yêu cầu.....	13
2.1 Xác định yêu cầu nghiệp vụ	13
2.1.1 Xác định các tác nhân	13
2.1.2 Xác định và mô tả các quy trình nghiệp vụ	13
2.1.3 Mô tả chi tiết quy trình nghiệp vụ.....	14
2.1.4 Sử dụng biểu đồ hoạt động để mô tả flow nghiệp vụ	18
2.2 Xác định yêu cầu chức năng	43
2.2.1 Xác định các yêu cầu mức cao.....	43
2.2.2 Xác định các Usecase.....	44
2.2.3 Khảo sát usecase	45
2.2.4 Sắp xếp ưu tiên các usecase	45
2.2.5 Biểu đồ usecase.....	46
2.2.6 Đặc tả usecase	50
2.3 Xác định yêu cầu phi chức năng	67
Phần 3. Phân tích	68
3.1 Phân tích tĩnh.....	68
3.1.1 Xác định lớp.....	68
3.1.2 Quan hệ giữa các lớp	68
3.1.3 Xây dựng biểu đồ lớp.....	71
3.2 Phân tích động	72

3.2.1	Hiện thực hóa usecase bằng biểu đồ tuần tự	72
3.2.2	Gán phương thức cho các lớp	91
Phần 4. Pha thiết kế.....	97	
4.1	Lựa chọn công nghệ mạng	97
4.1.1	Hoạt động của mô hình client-server	98
4.1.2	Ưu điểm của mô hình.....	98
4.1.3	Nhược điểm của mô hình.....	98
4.2	Xây dựng kiến trúc	99
4.2.1	Tổng quan về kiến trúc MVC	99
4.2.2	Tổng quan về kiến trúc MVVM.....	99
4.2.3	Sự khác nhau giữa MVC và MVVM	100
4.2.4	Lựa chọn kiến trúc	101
4.3	Thiết kế tương tranh và an toàn, bảo mật.....	102
4.3.1	Mã hóa thông tin nhạy cảm.....	102
4.3.2	Phòng chống Brute Force.....	103
4.4	Thiết kế logic.....	104
4.4.1	Thiết kế app client (mobile)	104
4.4.2	Thiết kế app quản trị (window).....	110
4.5	Thiết kế cơ sở dữ liệu	114
4.5.1	Chi tiết từng bảng.....	114
4.6	Phác họa giao diện người dùng	119
4.6.1	Giao diện khách hàng.....	119
4.6.2	Giao diện quản trị.....	124
4.7	Áp dụng design pattern: Mẫu Strategy vào xử lí quy trình thanh toán	131
4.7.1	Mục tiêu của mẫu Strategy	131
4.7.2	Vấn đề	131
4.7.3	Giải pháp	132
4.7.4	Cấu trúc của mẫu Strategy	132
4.7.5	Áp dụng vào xử lí quy trình thanh toán	133
4.8	Chọn ngôn ngữ/framework lập trình	134
4.8.1	Giới thiệu	134

4.8.2	Ưu điểm	134
4.8.3	Nhược điểm.....	135
Phần 5.	Lập trình ứng dụng	136
5.1	Nền tảng cơ sở dữ liệu (Firebase)	136
5.2	App quản trị (Windows).....	136
5.3	App client (Mobile).....	138
Phần 6.	Kết luận	141
6.1	Kết quả đồ án.....	141
6.2	Kỹ năng đạt được	141
6.3	Hướng phát triển	141
Tài liệu tham khảo		142

Danh sách hình vẽ

Figure 1 Đăng nhập.....	18
Figure 2 Đăng ký	19
Figure 3 Cập nhật thông tin người dùng	20
Figure 4 Xem thông tin phim.....	21
Figure 5 Xem hóa đơn	22
Figure 6 Đặt vé xem phim	23
Figure 7 Đặt đồ ăn.....	24
Figure 8 In vé	25
Figure 9 Thêm phim.....	26
Figure 10 Cập nhật thông tin phim	27
Figure 11 Xóa phim	28
Figure 12 Thêm rạp.....	29
Figure 13 Cập nhật thông tin rạp	30
Figure 14 Xóa rạp	31
Figure 15 Thêm đồ ăn	32
Figure 16 Cập nhật thông tin đồ ăn.....	33
Figure 17 Xóa đồ ăn.....	34
Figure 18 Thêm phòng chiếu	35
Figure 19 Cập nhật thông tin phòng chiếu.....	36
Figure 20 Xóa phòng chiếu.....	37
Figure 21 Thêm lịch chiếu	38
Figure 22 Cập nhật lịch chiếu	39
Figure 23 Xóa lịch chiếu.....	40
Figure 24 Cập nhật sơ đồ phòng chiếu	41
Figure 25 Phân quyền	42
Figure 26 Thông kê.....	43
Figure 27 Usecase tổng quan	46
Figure 28 Usecase khách hàng.....	47
Figure 29 Usecase nhân viên	48
Figure 30 Usecase admin	49
Figure 31 Các lớp người dùng và tài khoản.....	68
Figure 32 Các lớp cơ sở vật lý rạp chiếu	68
Figure 33 Các lớp liên quan đến phim.....	69
Figure 34 Các lớp sản phẩm bán trong rạp chiếu	69
Figure 35 Lớp lịch chiếu, vé xem phim	69
Figure 36 Các lớp liên quan đến hóa đơn	69
Figure 37 Lớp liên kết người dùng và mã giảm giá.....	70
Figure 38 Quan hệ giữa các lớp	70
Figure 39 Xây dựng biểu đồ lớp	71
Figure 40 Đăng nhập.....	72

Figure 41 Đăng ký	73
Figure 42 Chính sửa thông tin	74
Figure 43 Đặt vé xem phim	75
Figure 44 Đặt đồ ăn.....	76
Figure 45 Xem hóa đơn	77
Figure 46 Xem thông tin khuyến mãi	78
Figure 47 Xem chi tiết đồ ăn.....	78
Figure 48 Xem chi tiết phim	79
Figure 49 Tạo QR thanh toán.....	79
Figure 50 In vé	80
Figure 51 Thêm rạp	81
Figure 52 Cập nhật thông tin rạp	81
Figure 53 Xóa rạp	82
Figure 54 Thêm phòng chiếu	83
Figure 55 Cập nhật phòng chiếu	83
Figure 56 Xóa phòng chiếu.....	84
Figure 57 Thêm lịch chiếu	85
Figure 58 Cập nhật lịch chiếu	86
Figure 59 Xóa lịch chiếu.....	86
Figure 60 Cập nhật sơ đồ phòng chiếu	87
Figure 61 Xóa sơ đồ phòng chiếu	87
Figure 62 Phân quyền	88
Figure 63 Thông kê phim.....	89
Figure 64 Thông kê đồ ăn	89
Figure 65 Thông kê rạp	90
Figure 66 Gán phương thức cho biểu đồ lớp	91
Figure 67 Mô hình client-server	98
Figure 68 Cấu trúc của mẫu MVC [2]	99
Figure 69 Cấu trúc mẫu MVVM [2]	100
Figure 70 Hash config.....	103
Figure 71 Giới hạn đăng nhập khi request nhiều lần trong thời gian ngắn.....	103
Figure 72 Design class cho chức năng hiển thị trang chủ.....	104
Figure 73 Design class cho chức năng đăng nhập	104
Figure 74 Design class cho chức năng hiển thị menu.....	105
Figure 75 Design class cho chức năng đăng ký	106
Figure 76 Design class cho chức năng xem chi tiết phim.....	107
Figure 77 Design class cho chức năng chọn ghế	108
Figure 78 Design class cho chức năng thanh toán	108
Figure 79 Design class cho chức năng xem thông tin hóa đơn.....	109
Figure 80 Thiết kế logic form quản lý phim	110
Figure 81 Thiết kế logic form quản lý thể loại phim	110
Figure 82 Thiết kế logic form quản lý phương thức thanh toán	111

Figure 83 Thiết kế logic form quản lý rạp chiếu	111
Figure 84 Thiết kế logic form quản lý phòng chiếu.....	112
Figure 85 Thiết kế logic form quản lý sơ đồ phòng chiếu.....	112
Figure 86 Thiết kế logic form quản lý lịch chiếu.....	113
Figure 87 Thiết kế logic form quản lý giá vé.....	113
Figure 88 Sơ đồ dữ liệu quan hệ	114
Figure 89 Giao diện trang chủ	119
Figure 90 Trang thông tin chi tiết phim	119
Figure 91 Trang chọn lịch chiếu	120
Figure 92 Trang chọn chỗ ngồi	120
Figure 93 Trang thanh toán.....	121
Figure 94 Trang xem hóa đơn.....	121
Figure 95 Trang đăng ký	122
Figure 96 Trang đăng nhập	122
Figure 97 Menu.....	123
Figure 98 Trang quản lý rạp.....	124
Figure 99 Trang quản lý phim	125
Figure 100 Trang quản lý thể loại.....	126
Figure 101 Trang quản lý lịch chiếu	127
Figure 102 Trang quản lý phương thức thanh toán.....	128
Figure 103 Trang quản lý phòng chiếu	129
Figure 104 Trang quản lý sơ đồ phòng chiếu	130
Figure 105 Trang quản lý giá vé	131
Figure 106 Cấu trúc của mẫu Strategy	132
Figure 107 Áp dụng mẫu Strategy vào xử lý quy trình thanh toán.....	133
Figure 108 Thông kê phim có trong hệ thống.....	137
Figure 109 Chức năng tìm kiếm	137
Figure 110 Kiểm tra dữ liệu đầu vào trước khi thêm vào hệ thống.....	138
Figure 111 Cấu trúc project	139
Figure 112 Cấu trúc code từng trang giao diện.....	139
Figure 113 Trang chủ hiển thị danh sách phim, khuyến mãi.....	140
Figure 114 Giao diện chọn chỗ ngồi đặt vé	140

Bảng thuật ngữ

STT	Tiếng Anh	Tiếng Việt	Giải thích nội dung
1	Credit	Công trạng	Ghi công những người đóng góp cho bộ phim
2	Seat	Ghế ngồi	Ghế ngồi trong rạp, chứa thông tin về loại ghế
3	Ticket	Vé xem phim	Chứa thông tin về lịch chiếu, chỗ ngồi
4	Theater	Rạp chiếu	Chứa các thông tin cơ bản về rạp chiếu như vị trí, latitude, longitude, tên rạp
5	Actor/C	Diễn viên	Diễn viên có mặt trong phim
6	Movie	Phim	Phim chiếu trong rạp
7	Director	Đạo diễn	Đạo diễn sản xuất phim
8	Room	Phòng chiếu	Phòng chiếu phim trong rạp
9	Bill	Hóa đơn	Ghi thông tin sản phẩm, ngày tháng mua, khách hàng
10	Schedule	Lịch chiếu	Lịch chiếu cụ thể của phim tại rạp, vào ngày giờ
11	Room map	Sơ đồ phòng chiếu	Sơ đồ ghế và màn hình tại phòng chiếu
12	Genre	Thể loại phim	Thể loại mà phim thuộc về

Phần 1. Khảo sát

1.1 Ứng dụng CNTT vào quản lý rạp chiếu phim

Các rạp chiếu phim hiện nay thường được vận hành và quản lý với quy mô lớn và cách thức hiện đại, tích hợp nhiều công nghệ tiên tiến để đáp ứng nhu cầu ngày càng cao của khách hàng. Tuy nhiên, vẫn tồn tại một số vấn đề trong việc quản lý thông tin và dịch vụ khách hàng. Một rạp chiếu phim cần quản lý việc đặt vé, thanh toán của khách hàng, tình trạng nhân viên, lập lịch chiếu phim và bảo mật thông tin. Trong thời đại 4.0, việc quản lý rạp chiếu phim trên các thiết bị thông minh đang trở thành một xu hướng tất yếu. Công nghệ này không chỉ giúp quản lý và lưu trữ số lượng lớn thông tin trong thời gian dài mà còn hỗ trợ rạp chiếu phim tối ưu hóa hoạt động kinh doanh, nâng cao hiệu quả và sự hài lòng của khách hàng.

Qua khảo sát tình hình thực tế của các rạp chiếu phim trên địa bàn Hà Nội, em nhận thấy rằng việc quản lý rạp chiếu phim mặc dù đã hiện đại nhưng vẫn có những khía cạnh cần cải tiến. Việc áp dụng công nghệ thông tin hiện đại và xây dựng mô hình quản lý mới phù hợp, tiện lợi hơn là điều cần thiết. Những nghiệp vụ cụ thể cần được đề ra và xử lý khi xây dựng hệ thống bao gồm:

- Xây dựng hệ thống lưu trữ thông tin bán vé, thông tin phim một cách chính xác và nhanh chóng.
- Xây dựng hệ thống quản lý thông tin khách hàng, nhân viên, và nhà cung cấp nguyên vật liệu (thức ăn, đồ uống)
- Xây dựng hệ thống hỗ trợ quản lý tổng kết thu chi, lợi nhuận một cách tự động và chi tiết.
- Xây dựng hệ thống tự động hóa trong việc đặt vé, chọn chỗ ngồi và phục vụ khách hàng, giúp giảm thiểu thời gian chờ đợi và tăng cường trải nghiệm người dùng.
- Phát triển ứng dụng di động giúp khách hàng dễ dàng đặt vé, thanh toán trực tuyến và nhận thông tin về các chương trình khuyến mãi, lịch chiếu phim.

Việc áp dụng công nghệ thông tin vào quản lý rạp chiếu phim sẽ giúp nâng cao hiệu quả hoạt động, giảm thiểu rủi ro thất thoát, tạo ra một môi trường làm việc chuyên nghiệp và mang lại trải nghiệm tốt hơn cho khách hàng. Em hy vọng rằng với những cải tiến này, các rạp chiếu phim sẽ hoạt động hiệu quả hơn và phát triển bền vững trong tương lai.

1.2 Khảo sát hệ thống hiện tại của CGV

CGV là một trong năm Cụm Rạp Chiếu Phim lớn nhất toàn cầu và CGV Việt Nam là Nhà Phát Hành, nhà quản lý và vận hành Cụm Rạp Chiếu Phim CGV Cinemas lớn nhất tại Việt Nam.

1.2.1 Phạm vi khảo sát

Đề tài phân tích thiết kế hệ thống rạp chiếu phim này là một đề tài rất rộng, sơ lược qua chúng ta có thể kể một vài phần chính:

- Phần quản lý phim: nhập phim, lên lịch chiếu phim, marketing, ngưng chiếu phim....

- Phần quản lý đồ ăn: nhập nguyên liệu, quản lý nguyên liệu, chế biến món ăn, bán đồ ăn, thống kê kho....
- Phần quản lý rạp: quản lý cơ sở vật chất, kiểm tra bảo trì định kì, quản lý hợp đồng mặt bằng.... Đây là hệ thống quản lý chuỗi rạp chiếu trên toàn quốc.
- Phần quản lý phòng chiếu: quản lý sơ đồ phòng chiếu, quản lý vệ sinh, quản lý cơ sở vật chất, quản lý màn hình chiếu....
- Phần quản lý nhân sự: quản lý nhân viên, quản lý nhân sự vệ sinh, quản lý bảo vệ,.....
- Phần quản lý tài chính: thống kê, báo cáo, lên kế hoạch tài chính, đàm phán hợp đồng....

Có thể thấy đây là một đề tài rất rộng, bao quát nhiều mảng từ tài chính, nhân sự đến bài toán tối ưu, trong đề án này, em sẽ cố gắng phân tích thiết kế hệ thống chủ yếu về phía khách hàng, làm sao để khách hàng có được trải nghiệm tốt nhất, xây dựng hệ thống mềm dẻo, có cấu trúc rõ ràng, dễ dàng bảo trì và nâng cấp, đồng thời cũng sẽ đảm bảo tính bảo mật và an toàn thông tin tạo sự tin cậy tuyệt đối cho khách hàng.

1.2.2 Cơ cấu tổ chức

- Nhân viên: Các nhân viên sẽ đứng quầy để bán vé và đồ ăn. Họ sẽ tiếp nhận yêu cầu của khách hàng, hướng dẫn khách hàng chọn đồ, tiếp nhận thanh toán. Đôi khi là thực hiện một số bước xác nhận độ tuổi khách hàng.
- Admin: Vì không đi sâu vào khảo sát phía quản lý, em sẽ tạm gộp chung tất cả những chức vụ đăng sau quản lý phim, rạp, thống kê... vào 1 đối tượng gọi là admin. Đối tượng này sẽ chịu trách nhiệm quản lý các rạp, các phòng, phim, đồ ăn, thống kê, phân quyền...

1.2.3 Các quy trình nghiệp vụ chính

- **Đặt vé online**
Khách hàng sẽ sử dụng CGV mobile để tiến hành đặt vé. Trước tiên, CGV sẽ yêu cầu người dùng đăng nhập/đăng ký. Sau khi người dùng chọn phim, ứng dụng sẽ hiển thị danh sách ngày chiếu và rạp chiếu để người dùng chọn. Sau khi đã chọn ngày và rạp chiếu, ứng dụng tiếp tục hiển thị các suất chiếu. Sau khi chọn suất chiếu, người dùng sẽ được đưa tới trang chọn ghế ngồi, mỗi ghế ngồi được chọn, ứng dụng sẽ ngay lập tức hiển thị số tiền mà khách hàng phải trả. Sau khi chọn được chỗ ngồi, khách hàng sẽ được đưa tới trang thanh toán, tại đây khách hàng có thể sử dụng voucher, mua thêm đồ ăn, chọn phương thức thanh toán. Sau khi thanh toán thành công, CGV sẽ gửi email xác nhận đặt vé và mã QR để lấy vé tại quầy. Trong quá trình này, khách hàng có thể hoãn đặt vé ở bất kỳ bước nào trước khi thanh toán.
- **Đặt vé tại quầy**
Sau khi nhận tên phim và giờ chiếu muốn xem từ khách hàng, nhân viên sẽ hiển thị sơ đồ phòng chiếu của phim cho khách. Sau khi chọn xong chỗ ngồi, nhân viên sẽ khôi khách hàng phương thức thanh toán. Sau khi khách hàng thanh toán thành công, nhân viên sẽ in vé và giao vé cho khách hàng. Trong quá trình này, nhân viên có thể hoãn đặt vé ở bất kỳ bước nào trước khi thanh toán.
- **Quản lý phim**

Admin có thể thêm phim mới vào hệ thống sau khi nhập đủ các thông tin về phim như tên phim, poster phim, tóm tắt phim, ngôn ngữ, diễn viên, đạo diễn. Tương tự với đó là chỉnh sửa thông tin chi tiết phim, cũng như xóa phim khỏi hệ thống.

- Lên lịch chiếu

Admin có thể lên lịch chiếu phim. Điều này sẽ được lên lịch thủ công và hệ thống sẽ hỗ trợ phản hồi báo lỗi khi trùng lịch chiếu, lịch chiếu không hợp lệ..

1.3 Đề xuất hệ thống

Về mặt trải nghiệm khách hàng, CGV đã làm rất tốt, thật khó để một sinh viên có thể chỉ ra lỗi và khắc phục. Tuy nhiên, có một số hạn chế nhỏ ở ứng dụng CGV có thể kể đến như cung cấp thông tin về phim chưa đủ chi tiết. Ta có thể mở rộng bằng cách hiển thị nhiều thông tin của phim hơn như điểm số Tomato, IDBM, thông tin chi tiết hơn về diễn viên/đạo diễn.

Ngoài trừ hạn chế đó, CGV đã làm rất tốt về trải nghiệm khách hàng, thiết kế ứng dụng. Vì vậy trong báo cáo này, em sẽ cố gắng clone lại ứng dụng CGV mobile, chủ yếu vào tính năng đặt vé. Ngoài ra, em cũng sẽ tạo 1 ứng dụng Window để quản lý cơ sở dữ liệu dễ dàng cho việc kiểm thử.

Phần 2. Xác định yêu cầu

2.1 Xác định yêu cầu nghiệp vụ

2.1.1 Xác định các tác nhân

- Khách hàng: xem lịch chiếu, đặt vé, đặt đồ ăn, xem hóa đơn....
- Nhân viên: đặt vé, đặt đồ ăn tại quầy, tạo hóa đơn, in hóa đơn,...
- Admin: cập nhật phim, rạp, lịch chiếu, cập nhật nhân viên, thống kê doanh thu....
- Ngân hàng: thanh toán

2.1.2 Xác định và mô tả các quy trình nghiệp vụ

- 1 Đặt vé xem phim: khách hàng đặt vé xem phim thông qua ứng dụng hoặc đặt vé trực tiếp với nhân viên tại quầy
- 2 Đặt đồ ăn: khách hàng đặt đồ ăn thông qua ứng dụng hoặc đặt vé trực tiếp với nhân viên tại quầy
- 3 Xem hóa đơn đã mua: khách hàng xem hóa đơn đã mua qua hệ thống quản lý hóa đơn
- 4 Quản lý phim: sử dụng để thêm phim mới vào rạp, ngưng chiếu các phim ít khách xem
- 5 Quản lý lịch chiếu: admin thông qua các kế hoạch từ phòng ban thống kê để lên lịch chiếu phim, phòng chiếu phù hợp
- 6 Quản lý cơ sở hạ tầng: admin chỉnh sửa sơ đồ phòng chiếu, ghế trong phòng...
- 7 Báo cáo thống kê: lấy dữ liệu từ hệ thống để tạo các báo cáo, thống kê cho admin
- 8 Quản lý nhân sự: admin cập nhật, thêm nhân viên mới vào hệ thống, xóa nhân viên nghỉ việc khỏi hệ thống

2.1.3 Mô tả chi tiết quy trình nghiệp vụ

Đặt vé
<ol style="list-style-type: none">1. Khách hàng truy cập vào hệ thống đặt vé2. Khách hàng chọn phim muốn xem3. Khách hàng chọn rạp, giờ chiếu và ghế ngồi4. Hệ thống kiểm tra khách hàng đã đăng nhập chưa:<ol style="list-style-type: none">4.1 Nếu chưa, hệ thống chuyển hướng đến trang đăng nhập, yêu cầu đăng nhập4.2 Nếu rồi, hệ thống chuyển hướng đến trang thanh toán<ol style="list-style-type: none">4.2.1 Người dùng chọn phương thức thanh toán và thanh toán<ol style="list-style-type: none">4.2.1.1 Nếu thông tin hợp lệ, hệ thống ghi nhận giao dịch và gửi vé qua email cho người dùng4.2.1.2 Nếu không, hệ thống yêu cầu người dùng thử lại

Đặt đồ ăn
<ol style="list-style-type: none">1. Hệ thống kiểm tra khách hàng đã đăng nhập chưa:<ol style="list-style-type: none">1.1 Nếu chưa, hệ thống chuyển hướng đến trang đăng nhập, yêu cầu đăng nhập1.2 Nếu rồi, hệ thống chuyển hướng đến trang đặt đồ ăn<ol style="list-style-type: none">1.2.1 Khách hàng chọn đồ ăn1.2.2 Người dùng chọn phương thức thanh toán và thanh toán<ol style="list-style-type: none">1.2.2.1 Nếu thông tin hợp lệ, hệ thống ghi nhận giao dịch và gửi vé qua email cho người dùng1.2.2.2 Nếu không, hệ thống yêu cầu người dùng thử lại

Xem hóa đơn đã mua
<ol style="list-style-type: none">1. Hệ thống kiểm tra khách hàng đã đăng nhập chưa:<ol style="list-style-type: none">1.1 Nếu chưa, hệ thống chuyển hướng đến trang đăng nhập, yêu cầu đăng nhập1.2 Nếu rồi, hệ thống hiển thị danh sách hóa đơn đã mua<ol style="list-style-type: none">1.2.1.1 Nếu khách hàng nhấn vào 1 hóa đơn trong danh sách, hiển thị chi tiết hóa đơn

Quản lý phim (yêu cầu đăng nhập)

1. Thêm phim mới
 - 1.1 Người dùng nhập thông tin phim mới
 - 1.2 Hệ thống kiểm tra tính hợp lệ
 - 1.2.1 Nếu thông tin hợp lệ, thêm phim vào kho
 - 1.2.2 Nếu không hợp lệ, yêu cầu người dùng kiểm tra lại
2. Sửa thông tin phim
 - 2.1 Người dùng tìm kiếm thông tin phim để lấy thông tin cũ
 - 2.2 Người dùng nhập thông tin chỉnh sửa
 - 2.3 Hệ thống kiểm tra tính hợp lệ
 - 2.3.1 Nếu thông tin hợp lệ, thêm phim vào kho
 - 2.3.2 Nếu không hợp lệ, yêu cầu người dùng kiểm tra lại
3. Xóa phim
 - 3.1 Người dùng nhập id phim
 - 3.2 Hệ thống kiểm tra tồn tại
 - 3.2.1 Nếu tồn tại, hệ thống hiển thị cảnh báo, yêu cầu người dùng xác nhận
 - 3.2.2 Nếu không, yêu cầu người dùng kiểm tra lại

Quản lý lịch chiếu (yêu cầu đăng nhập)

1. Thêm lịch chiếu mới
 - 1.1 Người dùng nhập thông tin lịch chiếu mới
 - 1.2 Hệ thống kiểm tra tính hợp lệ
 - 1.2.1 Nếu thông tin hợp lệ, thêm lịch chiếu vào kho
 - 1.2.2 Nếu không hợp lệ, yêu cầu người dùng kiểm tra lại
2. Sửa thông tin lịch chiếu
 - 2.1 Người dùng tìm kiếm thông tin lịch chiếu để lấy thông tin cũ
 - 2.2 Người dùng nhập thông tin chỉnh sửa
 - 2.3 Hệ thống kiểm tra tính hợp lệ
 - 2.3.1 Nếu thông tin hợp lệ, cập nhật lịch chiếu vào kho
 - 2.3.2 Nếu không hợp lệ, yêu cầu người dùng kiểm tra lại
3. Xóa lịch chiếu
 - 3.1 Người dùng nhập thông tin lịch chiếu (ngày, phim, rạp, phòng, khung giờ)
 - 3.2 Hệ thống kiểm tra tồn tại
 - 3.2.1 Nếu tồn tại, hệ thống hiển thị cảnh báo, yêu cầu người dùng xác nhận
 - 3.2.2 Nếu không, yêu cầu người dùng kiểm tra lại

Quản lý nhân sự (yêu cầu đăng nhập)

1. Thêm nhân viên mới
 - 1.1 Nhân viên điền form đăng ký, thông tin được gửi lên hệ thống
 - 1.2 Admin kiểm tra thông tin và phê duyệt yêu cầu, hoặc xóa yêu cầu nếu không đồng ý
2. Sửa thông tin nhân viên
 - 2.1 Nhân viên truy cập trang chỉnh sửa thông tin, chỉnh sửa thông tin
 - 2.2 Hệ thống kiểm tra tính hợp lệ
 - 2.2.1 Nếu thông tin hợp lệ, phê duyệt yêu cầu chỉnh sửa
 - 2.2.2 Nếu không hợp lệ, yêu cầu người dùng kiểm tra lại
3. Xóa nhân viên
 - 3.1 Admin nhập id phim
 - 3.2 Hệ thống kiểm tra tồn tại
 - 3.2.1 Nếu tồn tại, hệ thống hiển thị cảnh báo, yêu cầu người dùng xác nhận
 - 3.2.2 Nếu không, yêu cầu người dùng kiểm tra lại

Quản lý cơ sở hạ tầng (yêu cầu đăng nhập)

1. Thêm rạp mới
 - 1.1 Người dùng nhập thông tin rạp mới
 - 1.2 Hệ thống kiểm tra tính hợp lệ
 - 1.2.1 Nếu thông tin hợp lệ, thêm rạp vào kho
 - 1.2.2 Nếu không hợp lệ, yêu cầu người dùng kiểm tra lại
2. Sửa thông tin phim
 - 2.1 Người dùng tìm kiếm thông tin rạp để lấy thông tin cũ
 - 2.2 Người dùng nhập thông tin chỉnh sửa
 - 2.3 Hệ thống kiểm tra tính hợp lệ
 - 2.3.1 Nếu thông tin hợp lệ, thêm rạp vào kho
 - 2.3.2 Nếu không hợp lệ, yêu cầu người dùng kiểm tra lại
3. Xóa rạp
 - 3.1 Người dùng nhập id rạp
 - 3.2 Hệ thống kiểm tra tồn tại
 - 3.2.1 Nếu tồn tại, hệ thống hiển thị cảnh báo, yêu cầu người dùng xác nhận
 - 3.2.2 Nếu không, yêu cầu người dùng kiểm tra lại
4. Thêm phòng chiếu
 - 4.1 Người dùng nhập thông tin phòng chiếu mới, cùng với file excel sơ đồ ghế trong phòng
 - 4.2 Hệ thống kiểm tra tính hợp lệ
 - 4.2.1 Nếu thông tin hợp lệ, thêm phòng chiếu vào kho
 - 4.2.2 Nếu không hợp lệ, yêu cầu người dùng kiểm tra lại
5. Sửa phòng chiếu

- 5.1 Người dùng tìm kiếm thông tin rạp để lấy thông tin cũ
- 5.2 Người dùng nhập thông tin chỉnh sửa
- 5.3 Hệ thống kiểm tra tính hợp lệ
 - 5.3.1 Nếu thông tin hợp lệ, cập nhật phòng chiếu vào kho
 - 5.3.2 Nếu không hợp lệ, yêu cầu người dùng kiểm tra lại
- 6. Xóa phòng chiếu
 - 6.1 Người dùng nhập id phòng chiếu
 - 6.2 Hệ thống kiểm tra tồn tại
 - 6.2.1 Nếu tồn tại, hệ thống hiển thị cảnh báo, yêu cầu người dùng xác nhận
 - 6.2.2 Nếu không, yêu cầu người dùng kiểm tra lại

Báo cáo thống kê (yêu cầu đăng nhập)

- 1. Người dùng yêu cầu xem thống kê
- 2. Hệ thống lấy dữ liệu và hiển thị thống kê
- 3. Người dùng chọn các filter
- 4. Hệ thống hiển thị thống kê theo filter

2.1.4 Sử dụng biểu đồ hoạt động để mô tả flow nghiệp vụ

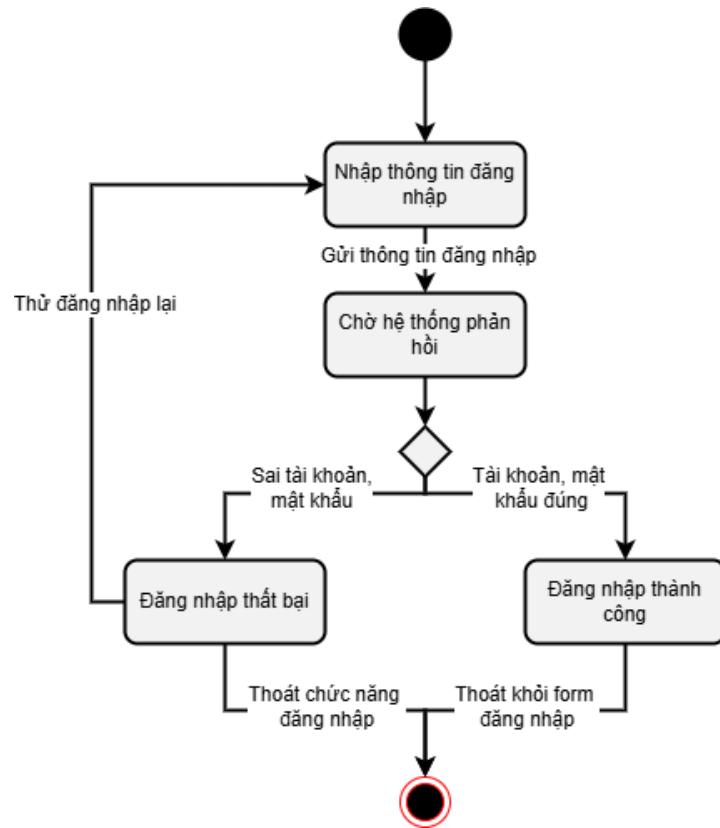


Figure 1 Đăng nhập

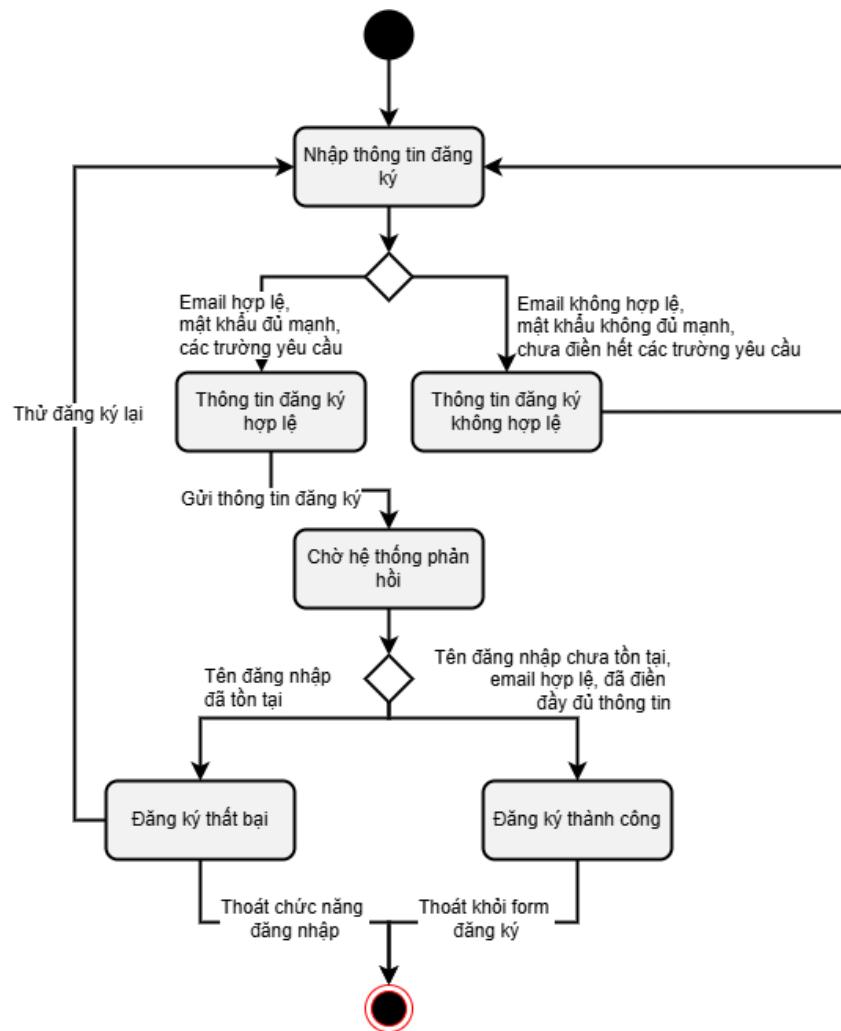


Figure 2 Đăng ký

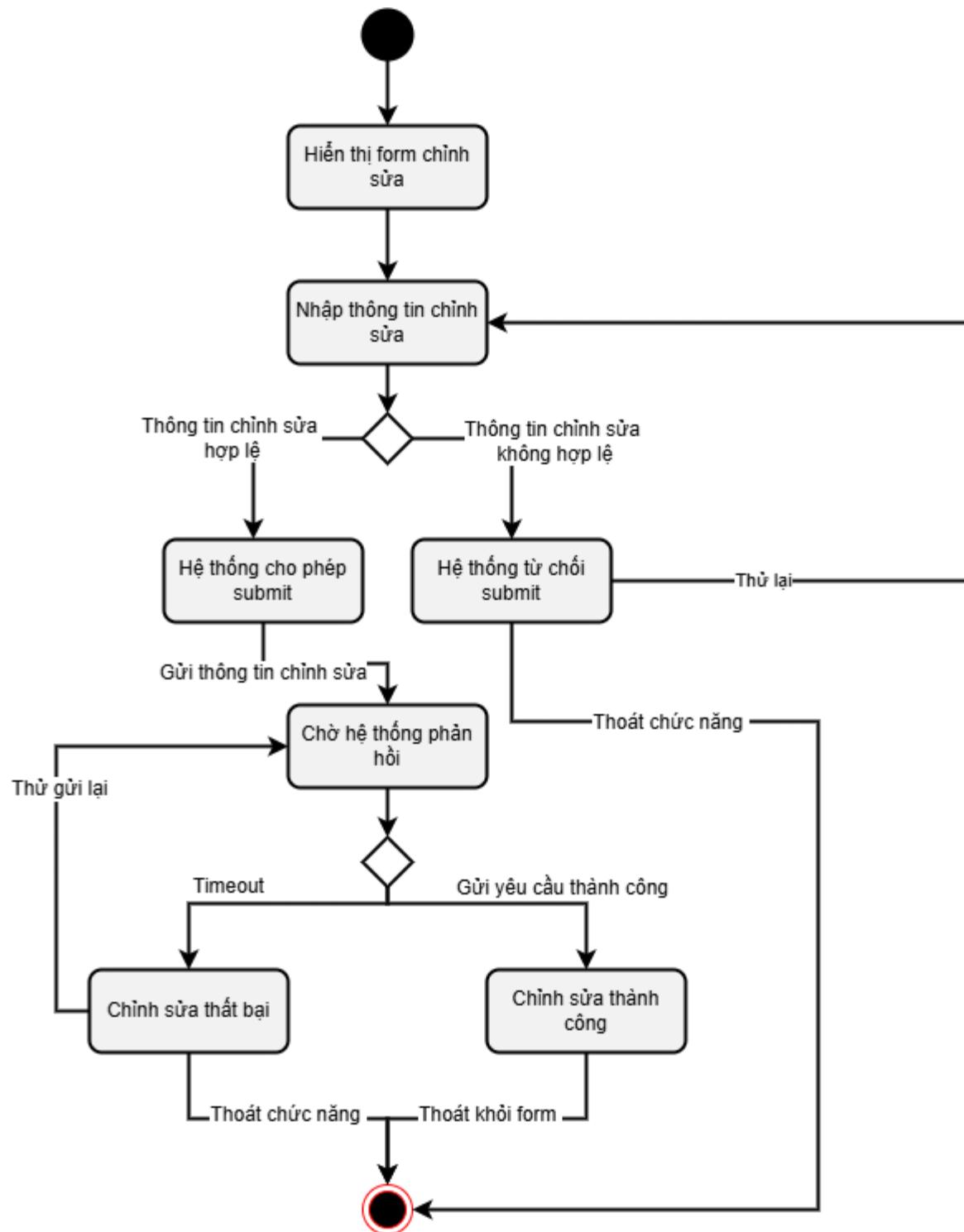


Figure 3 Cập nhật thông tin người dùng

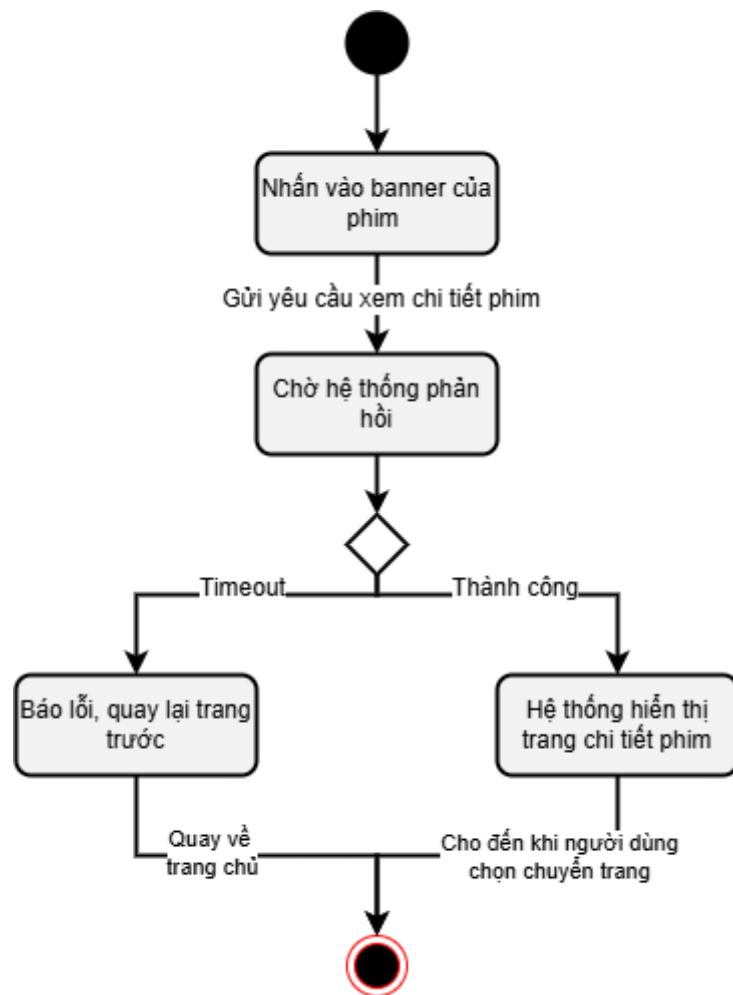


Figure 4 Xem thông tin phim

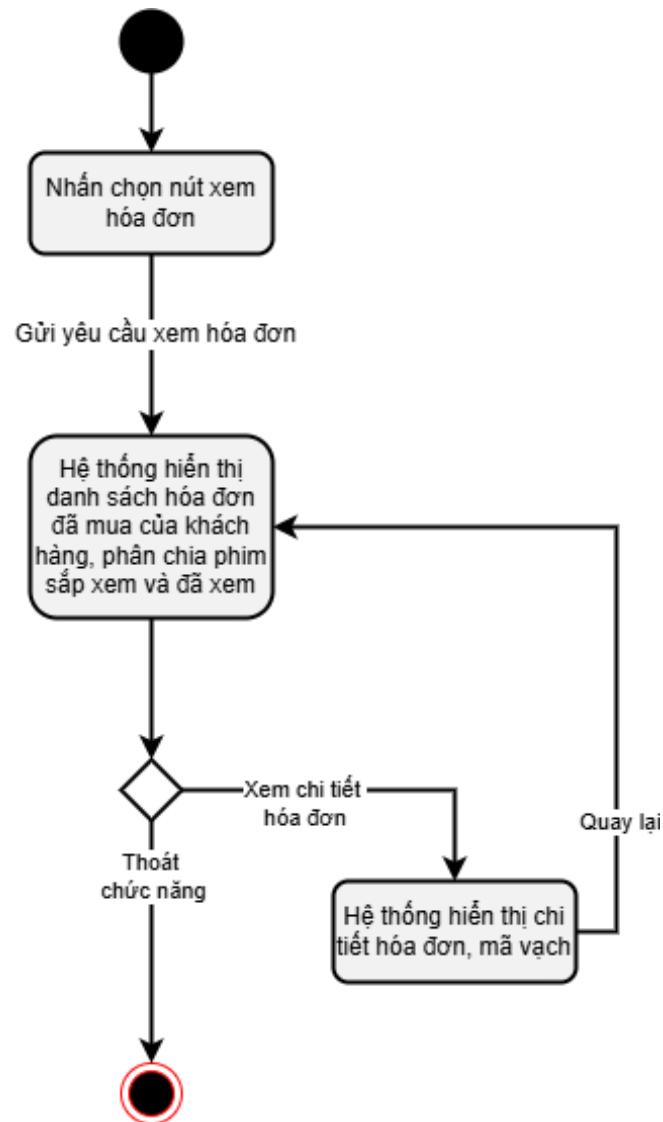


Figure 5 Xem hóa đơn

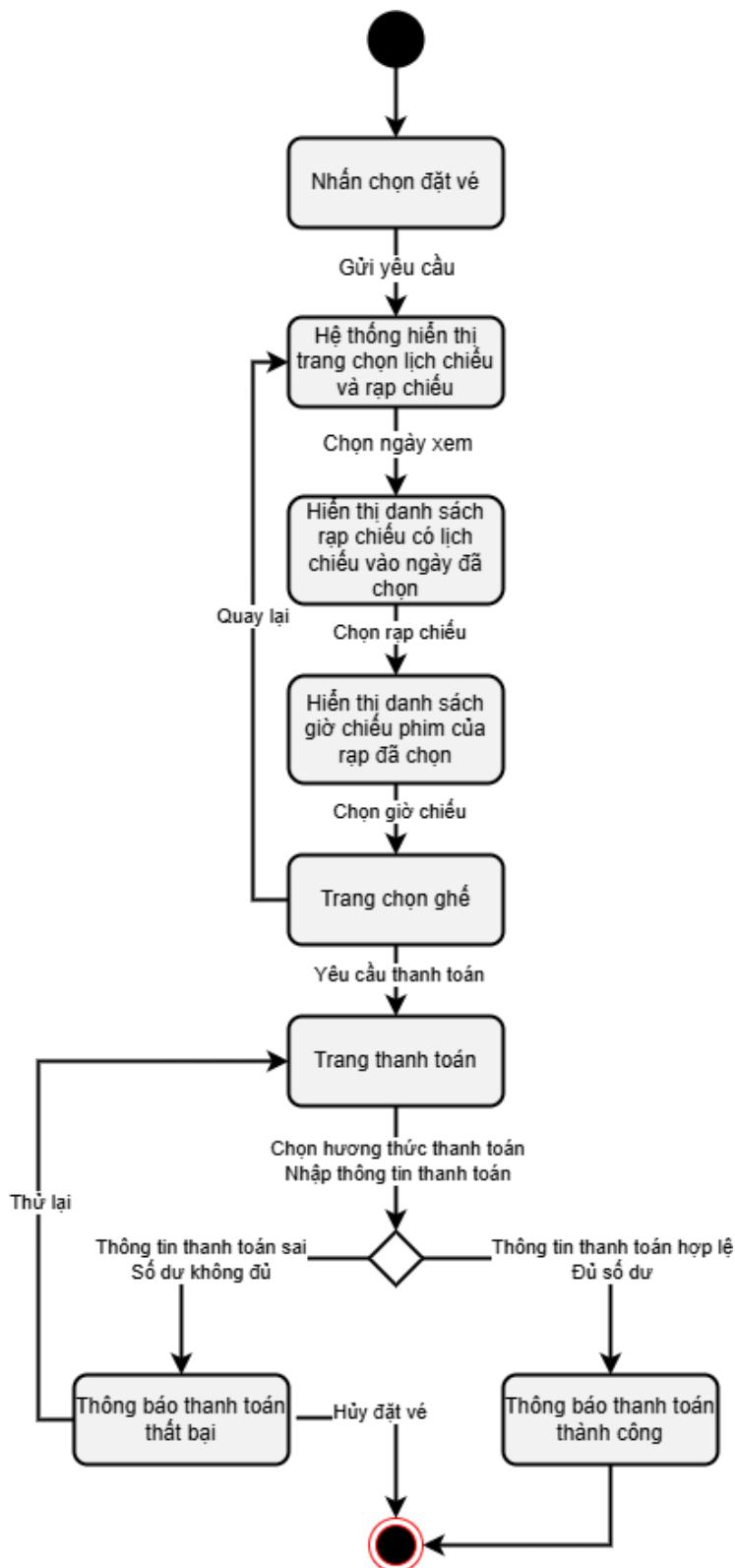


Figure 6 Đặt vé xem phim

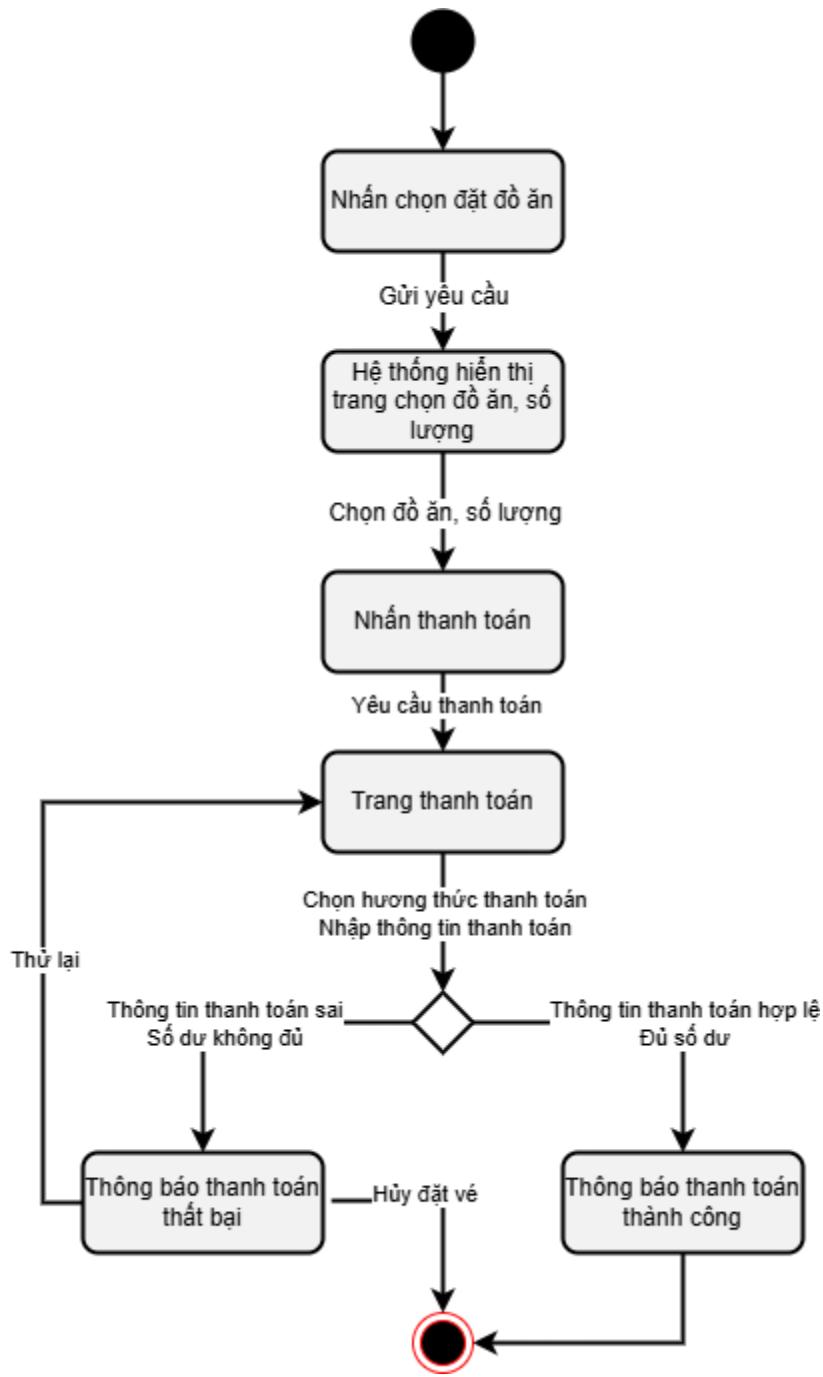


Figure 7 Đặt đồ ăn

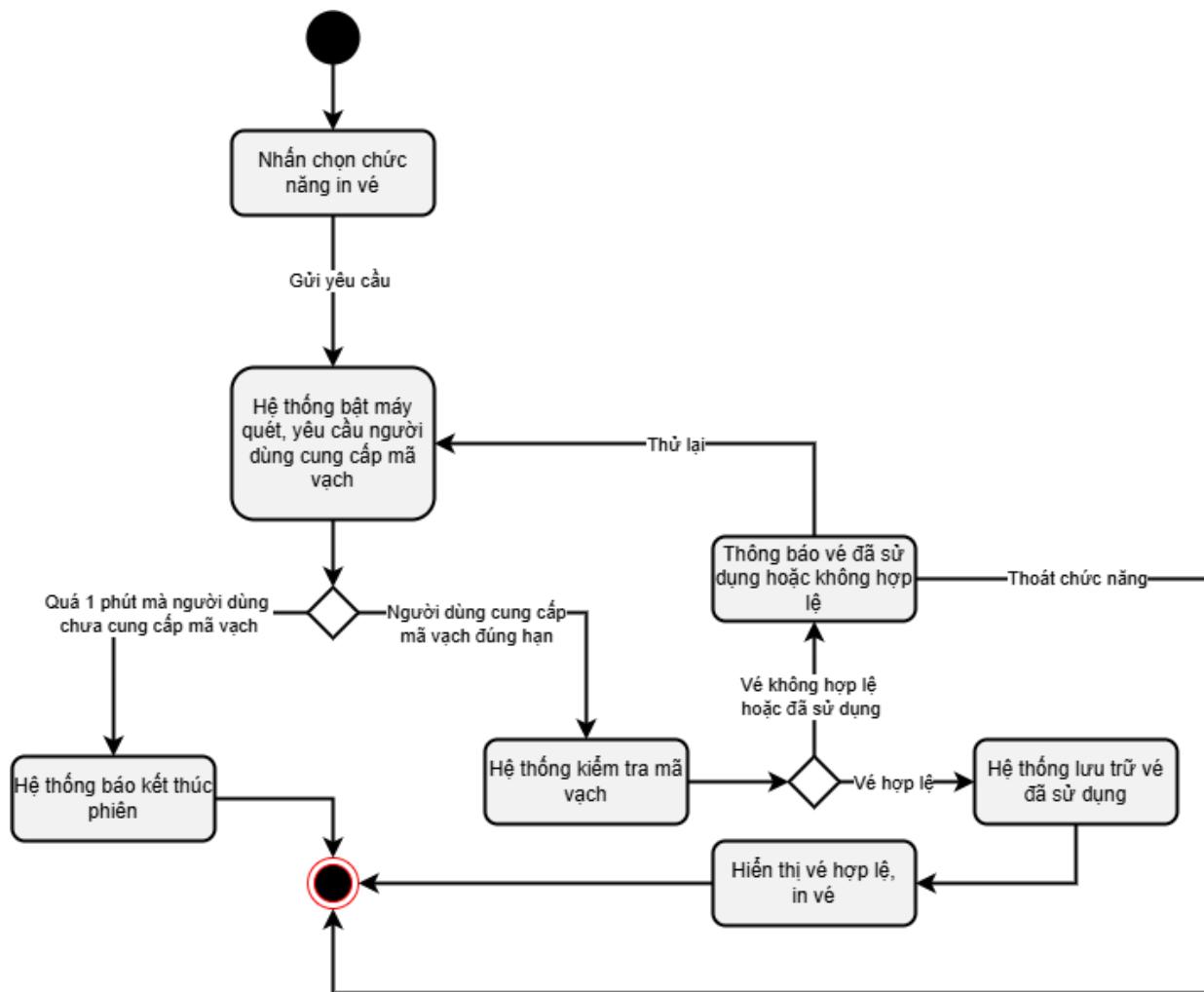


Figure 8 In vé

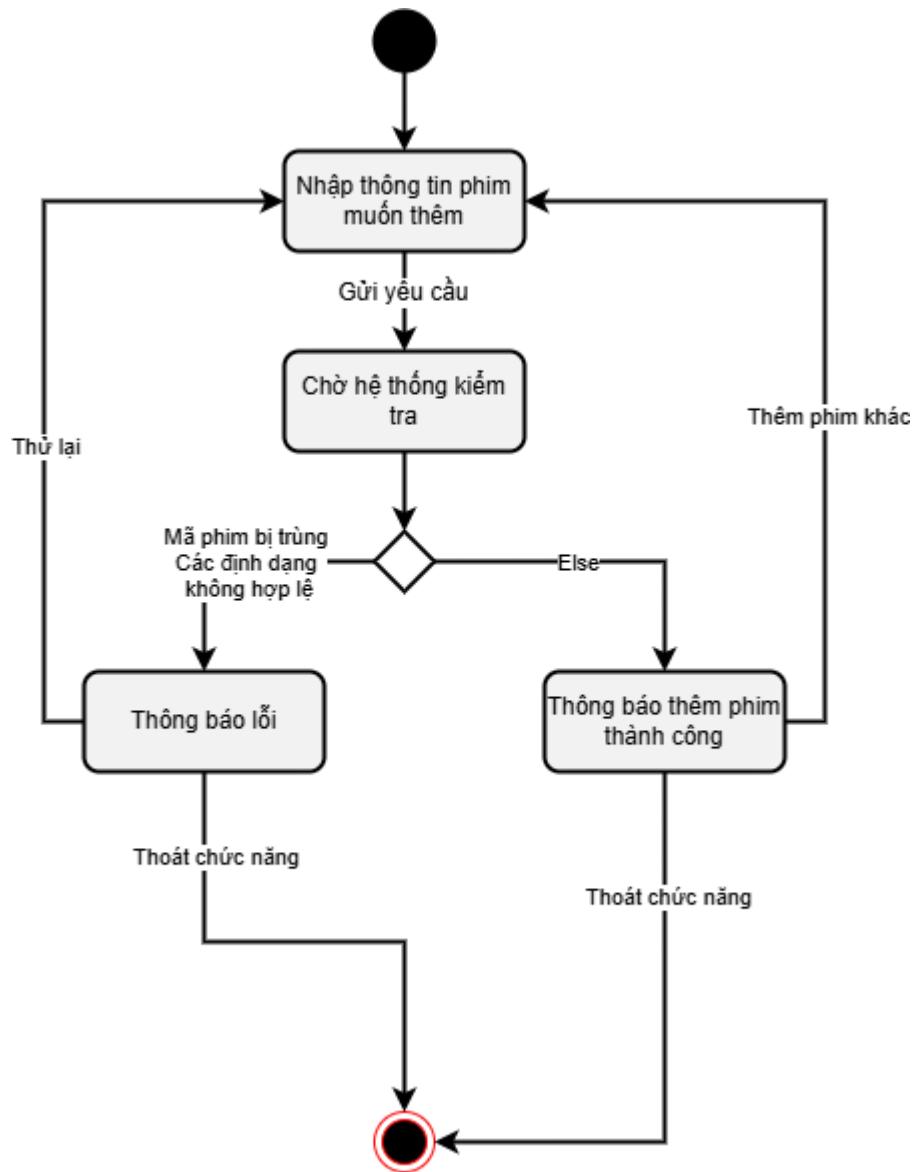


Figure 9 Thêm phim

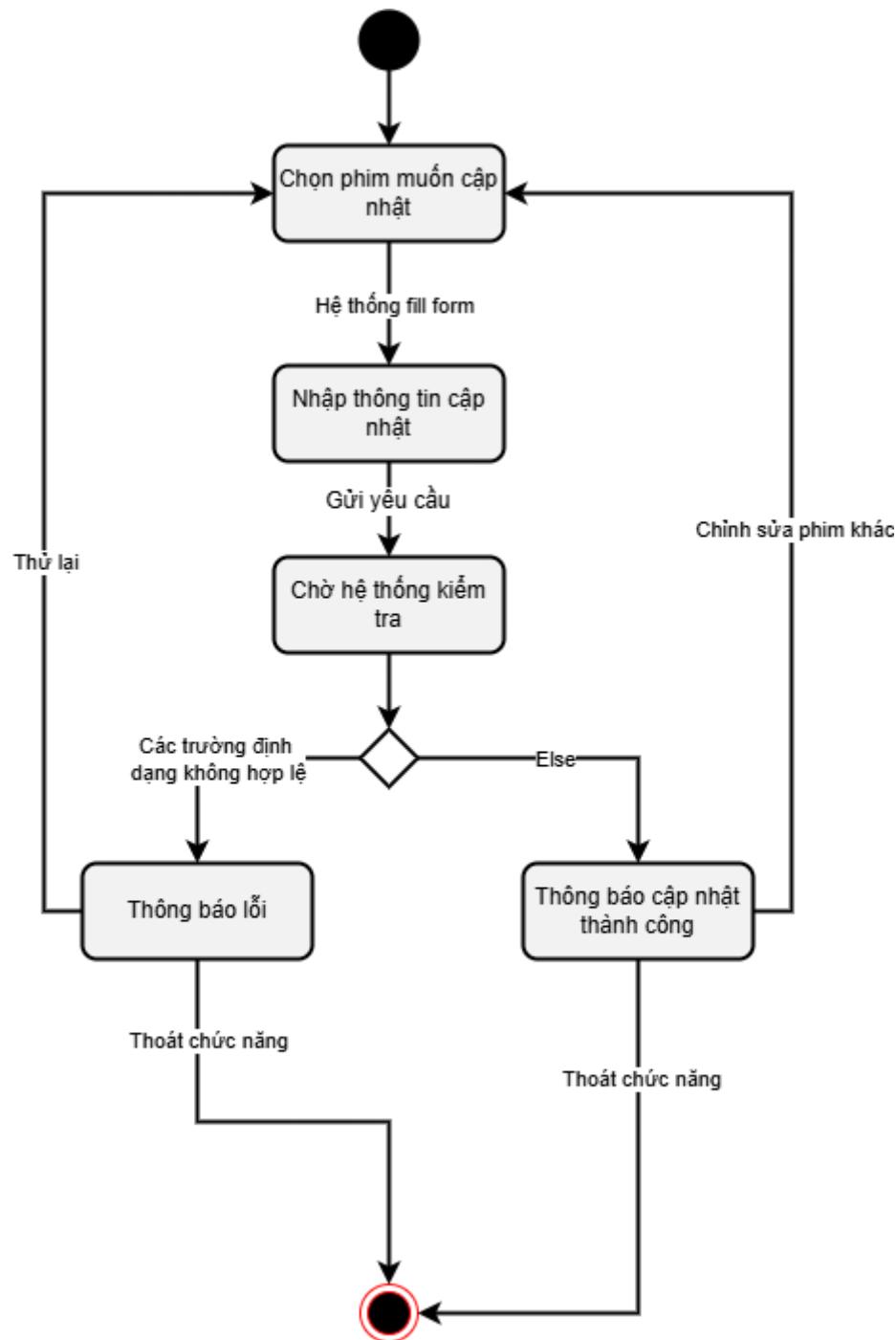


Figure 10 Cập nhật thông tin phim

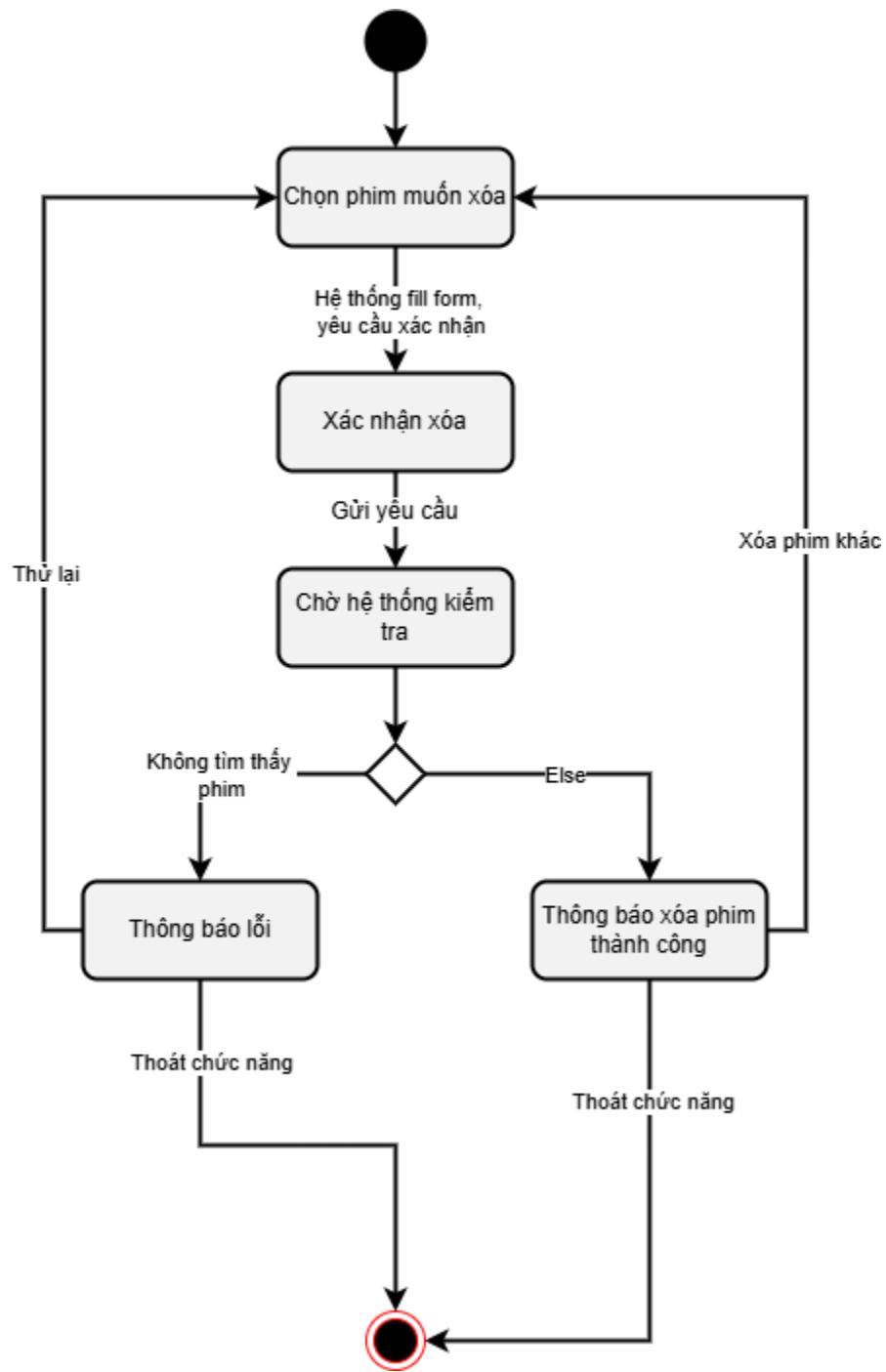


Figure 11 Xóa phim

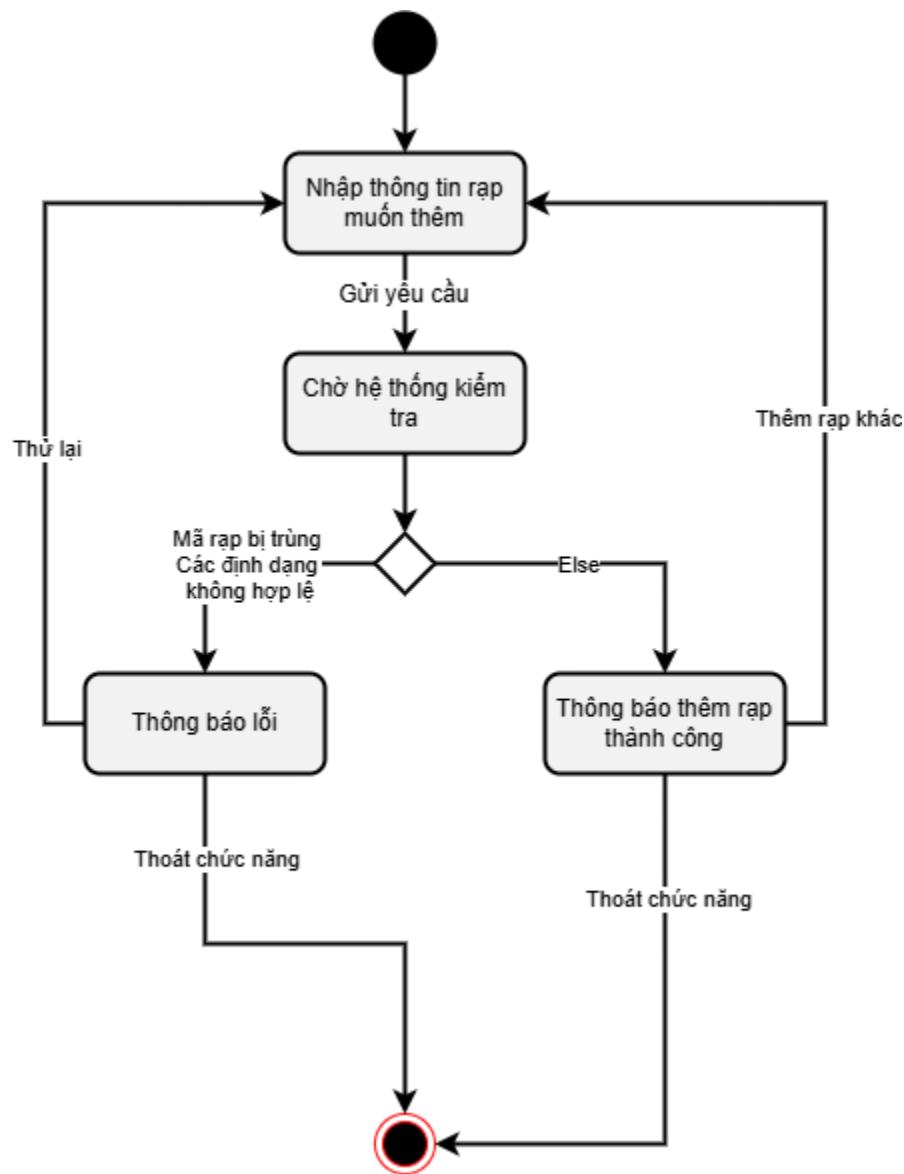


Figure 12 Thêm rạp

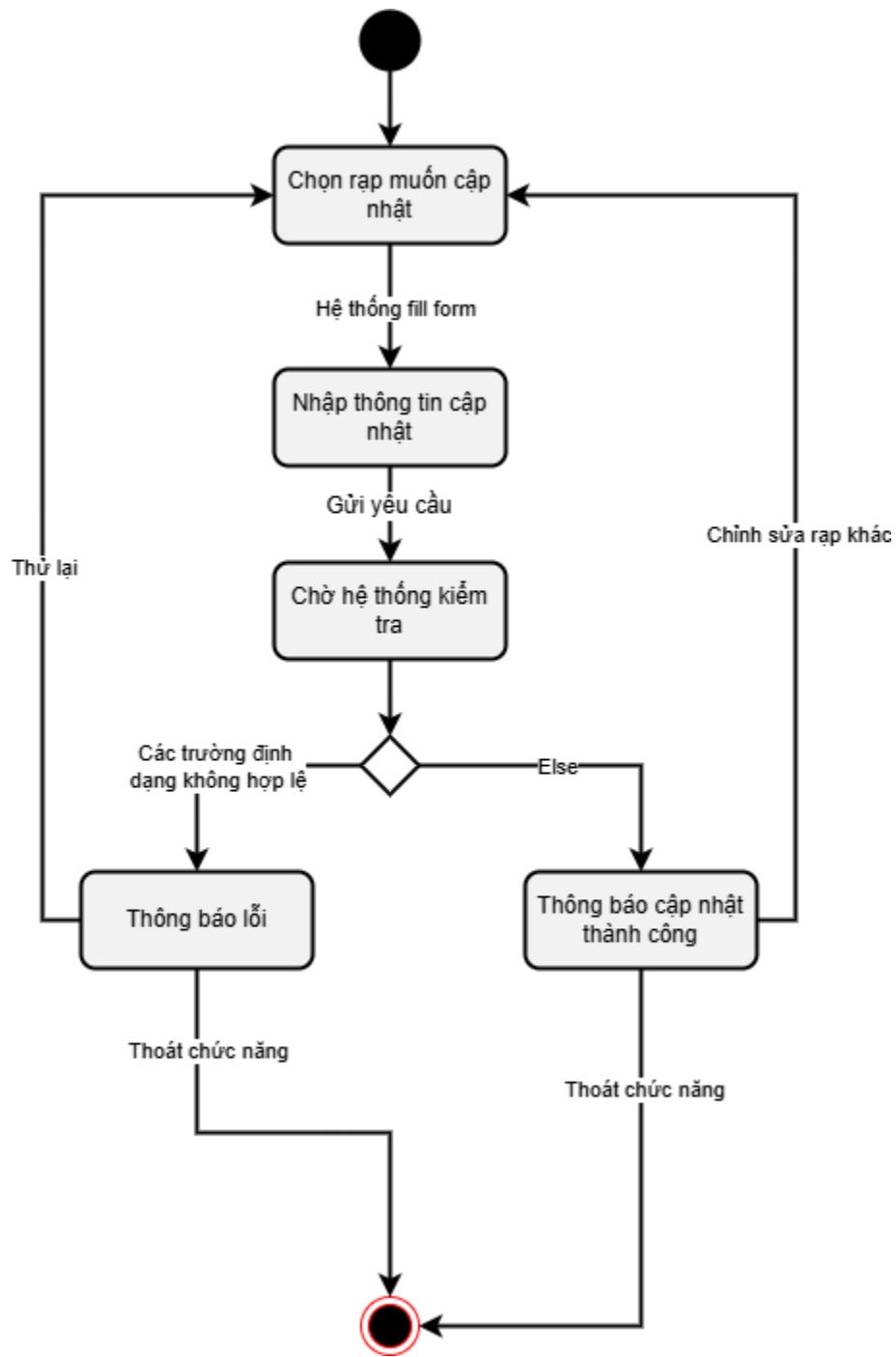


Figure 13 Cập nhật thông tin rạp

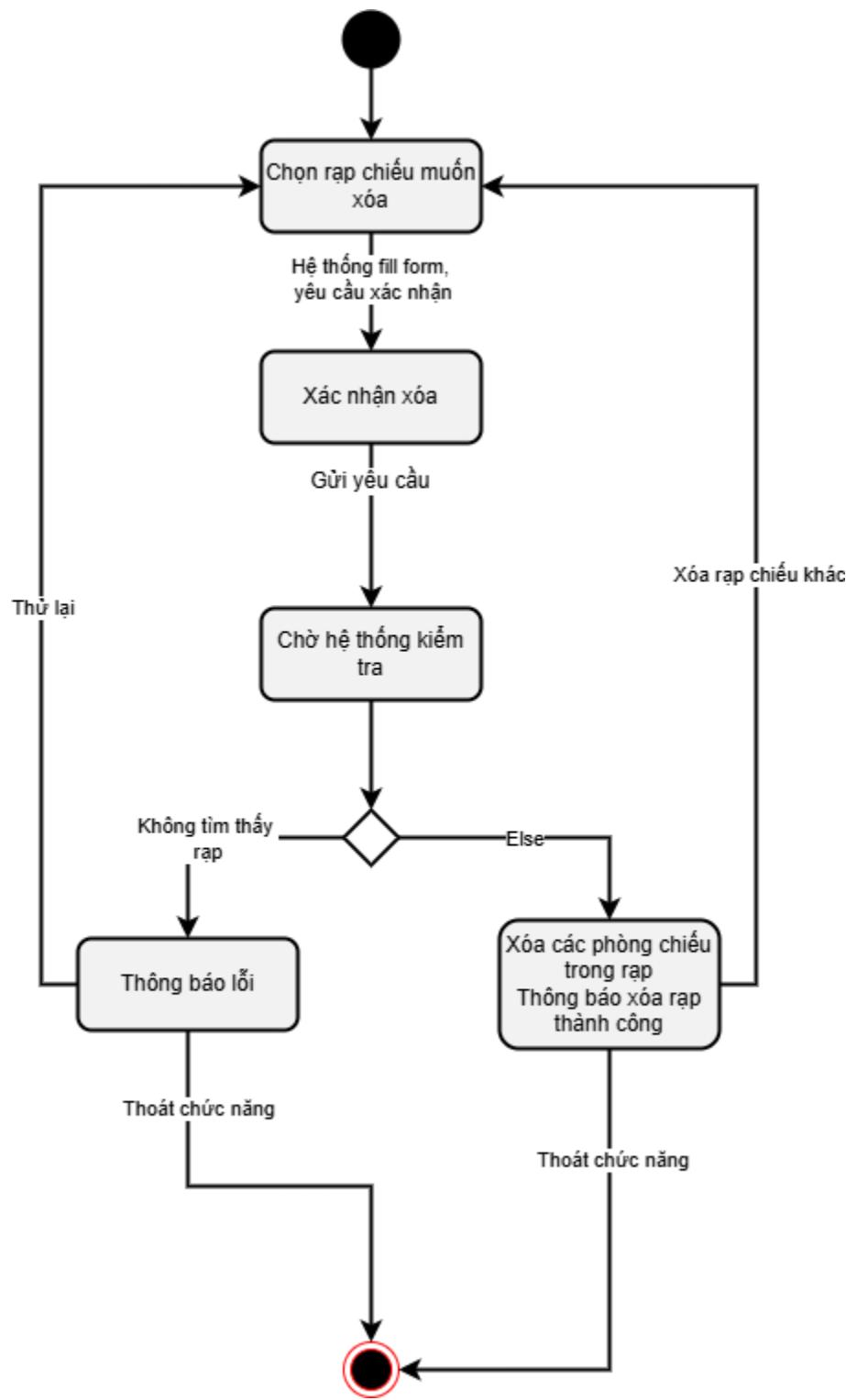


Figure 14 Xóa rạp

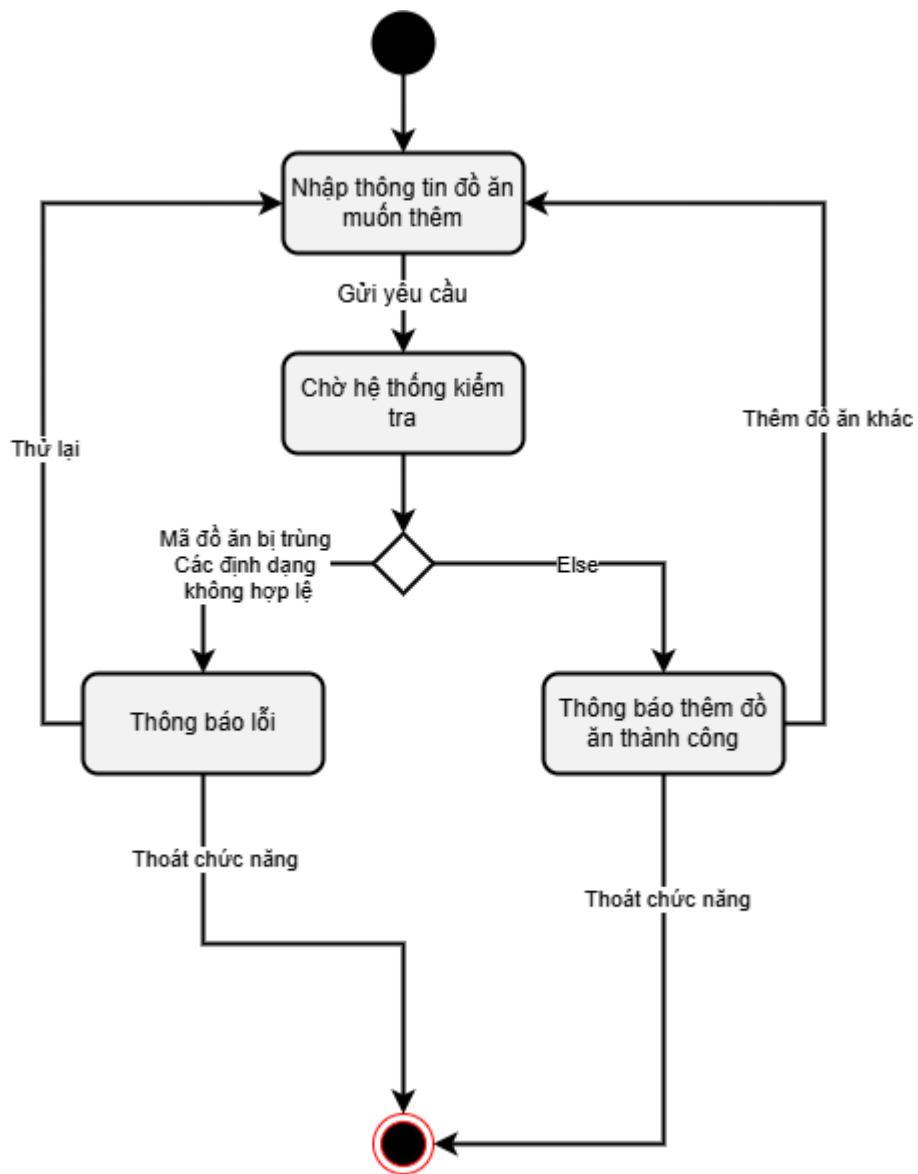


Figure 15 Thêm đồ ăn

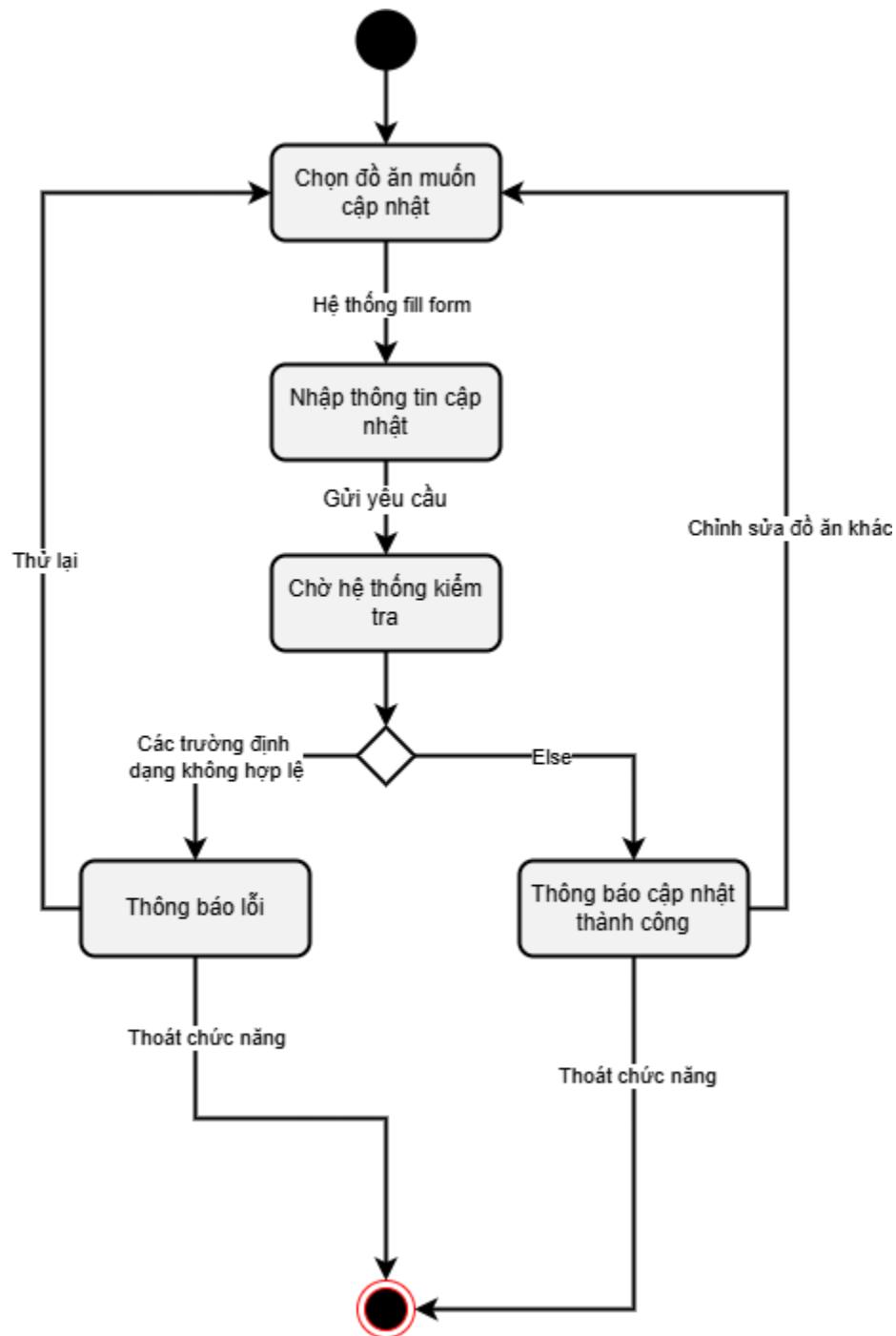


Figure 16 Cập nhật thông tin đồ ăn

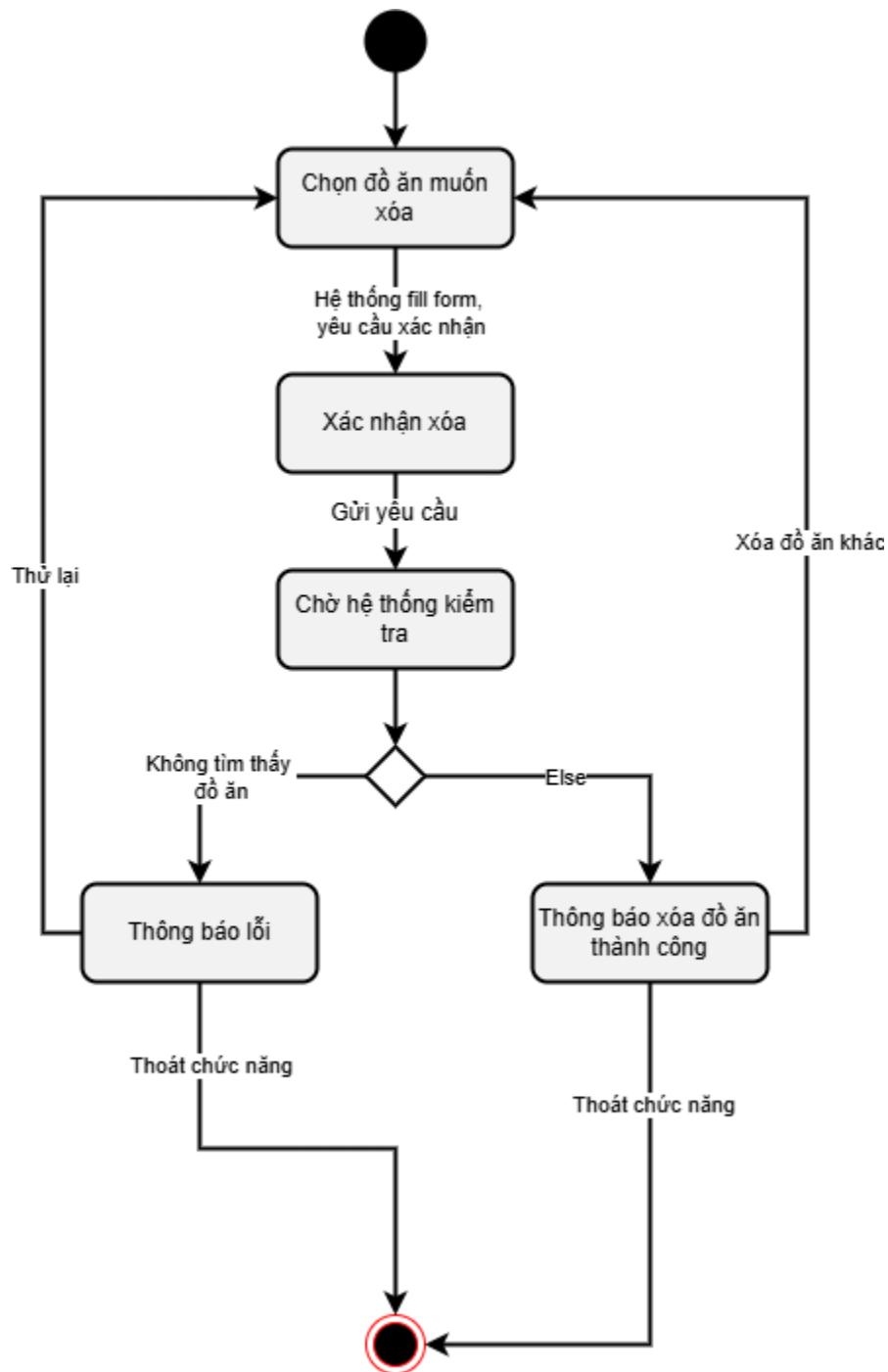


Figure 17 Xóa đồ ăn

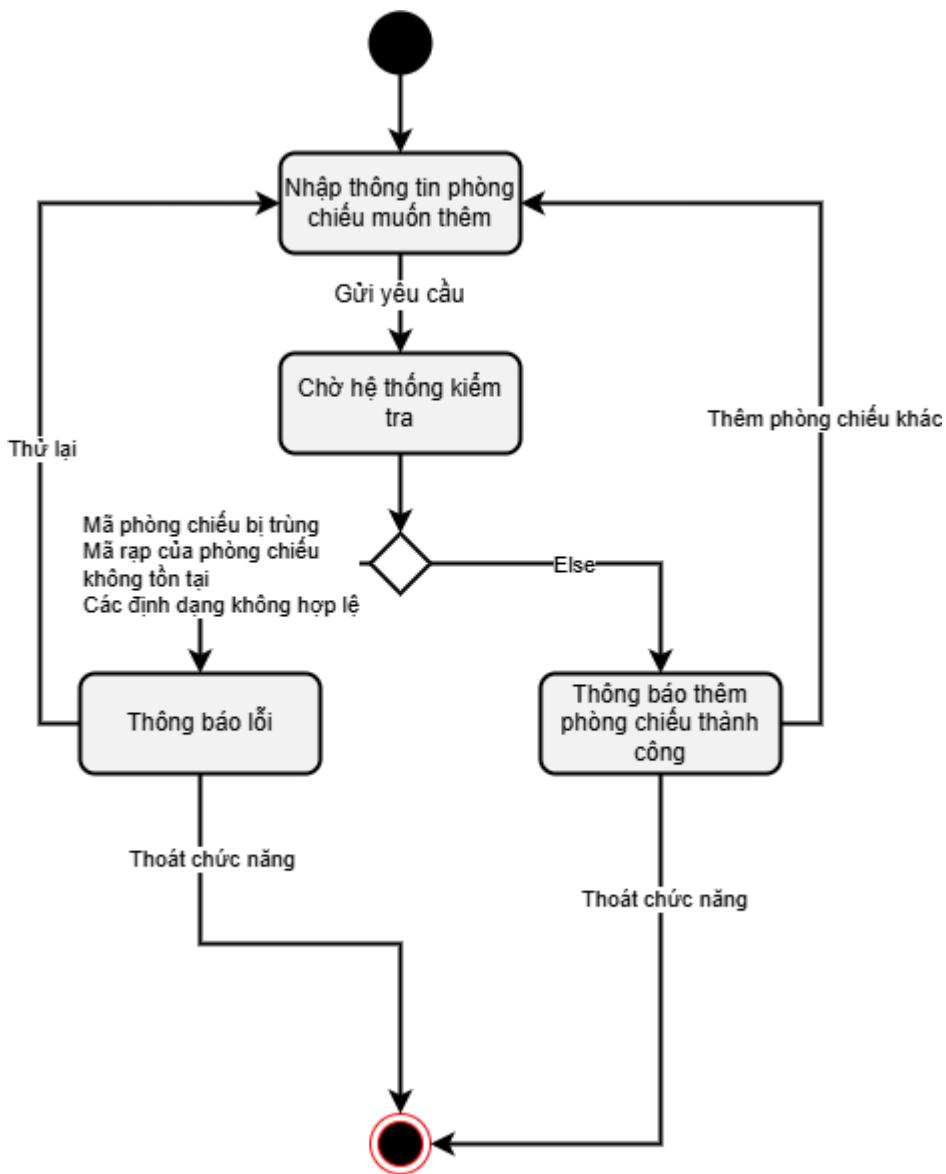


Figure 18 Thêm phòng chiếu

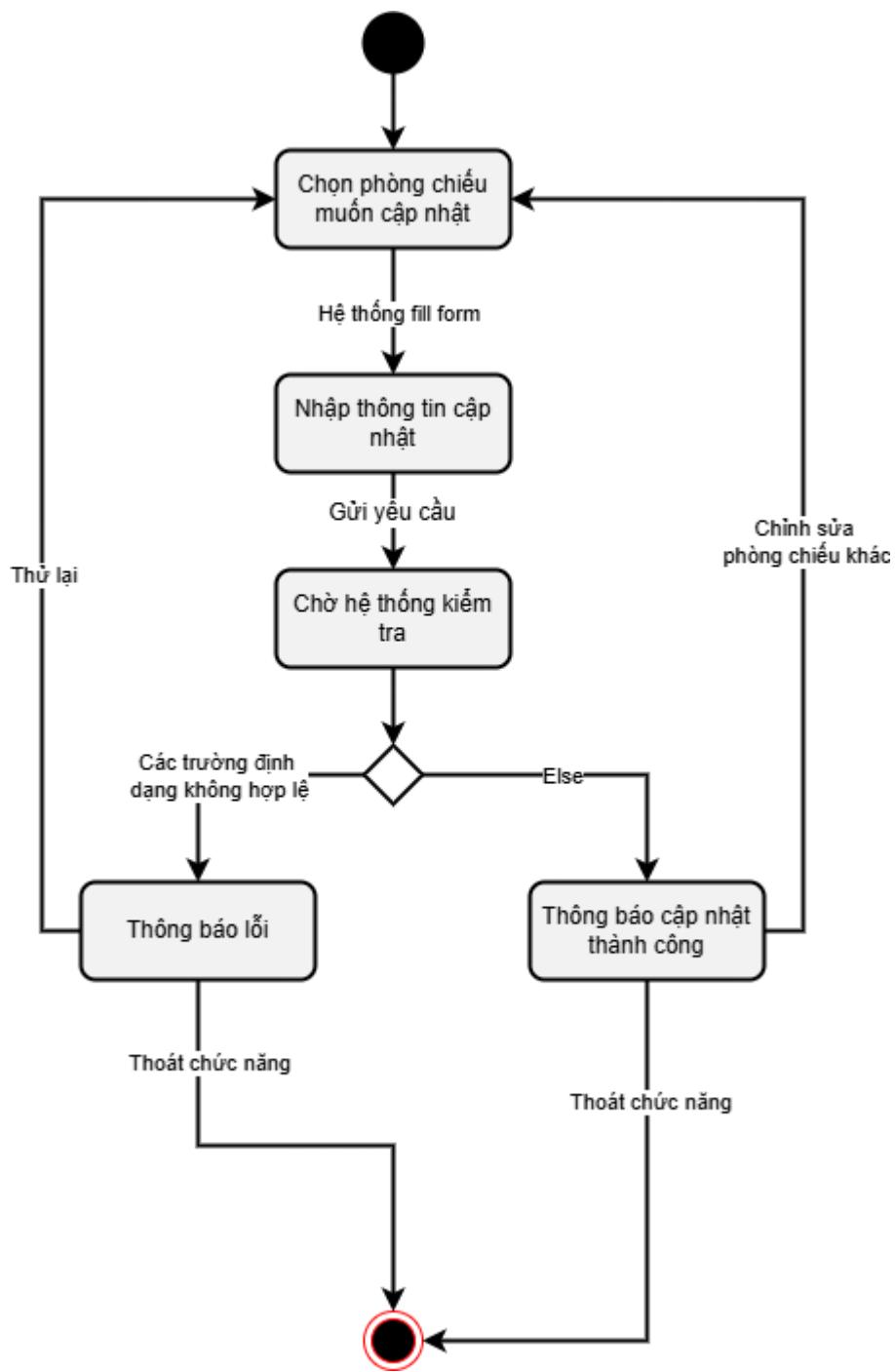


Figure 19 Cập nhật thông tin phòng chiếu

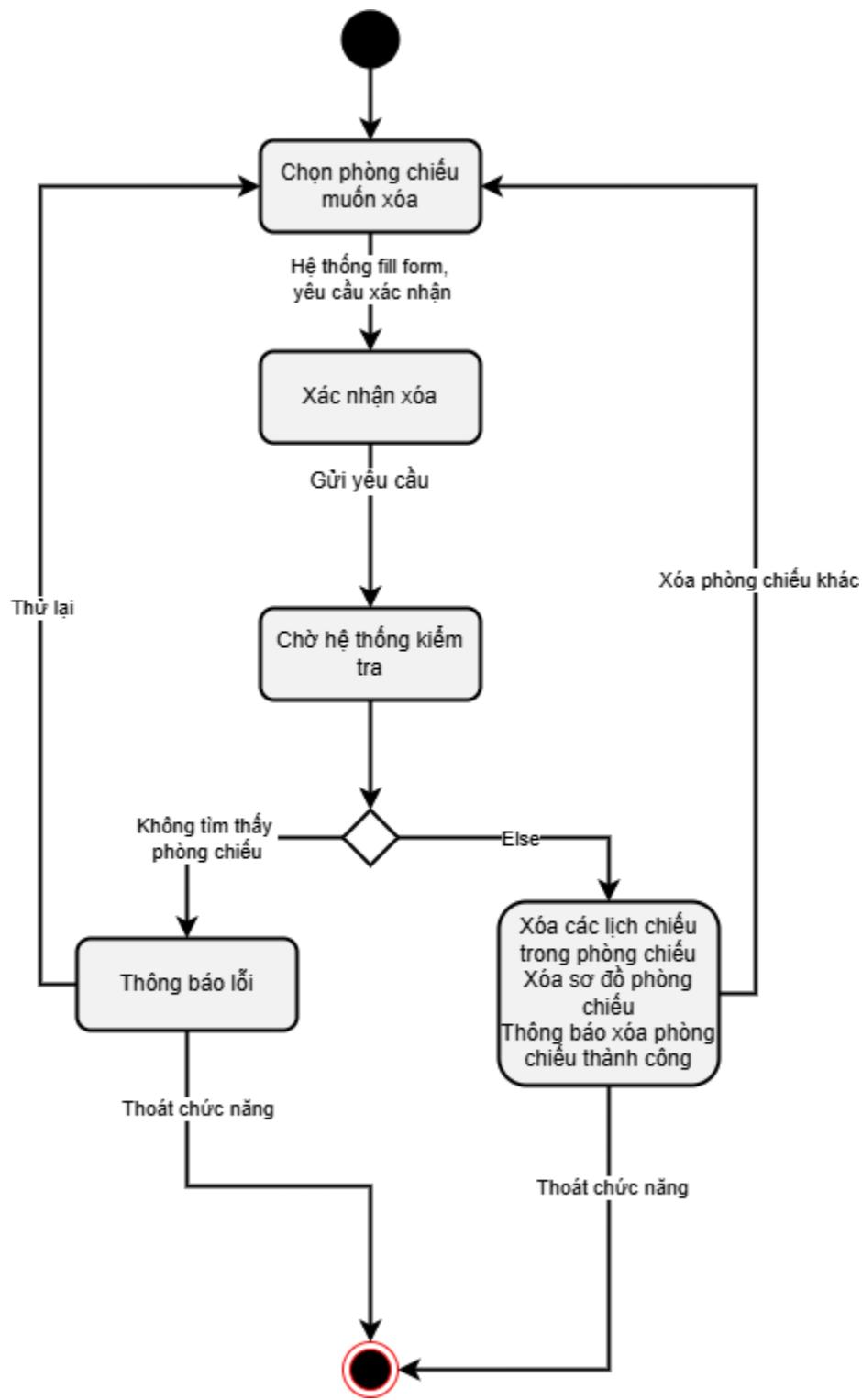


Figure 20 Xóa phòng chiếu

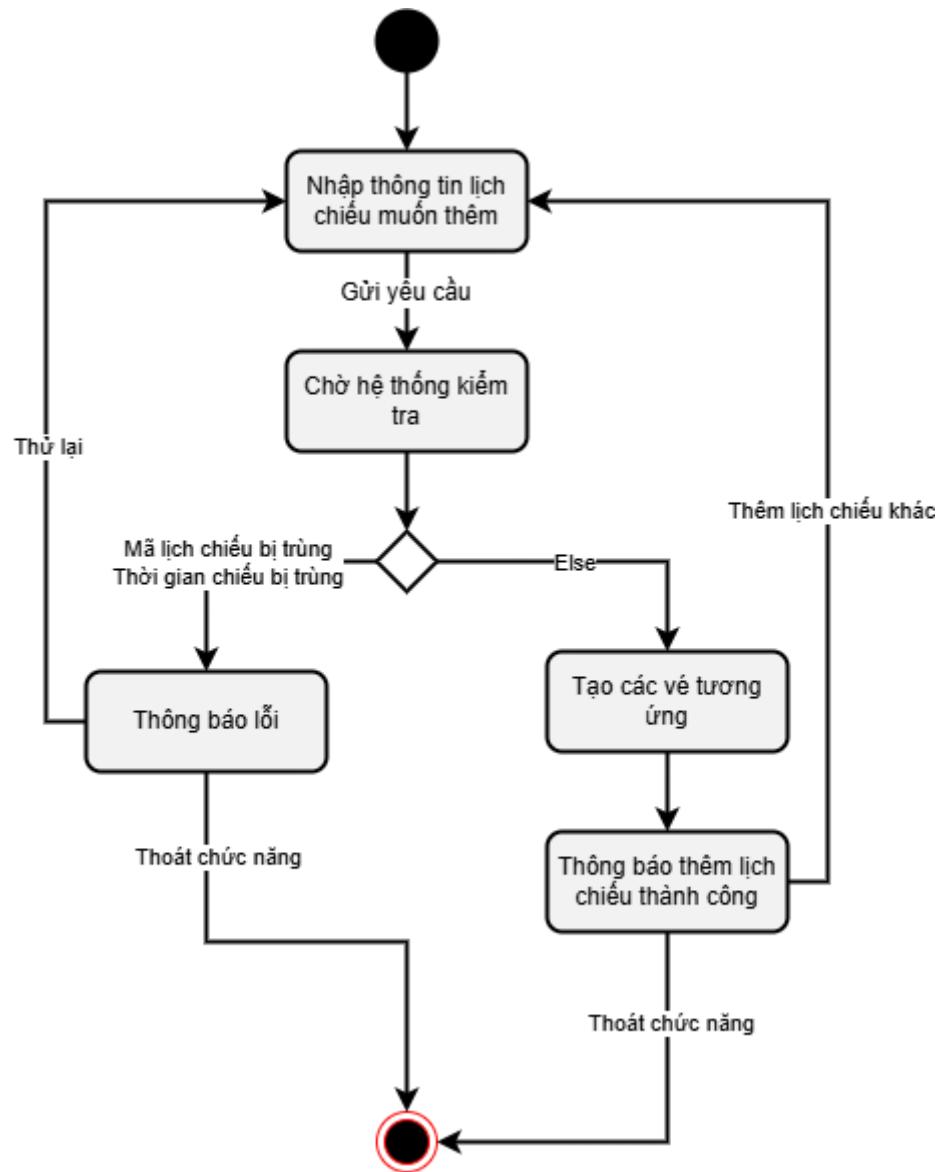


Figure 21 Thêm lịch chiếu

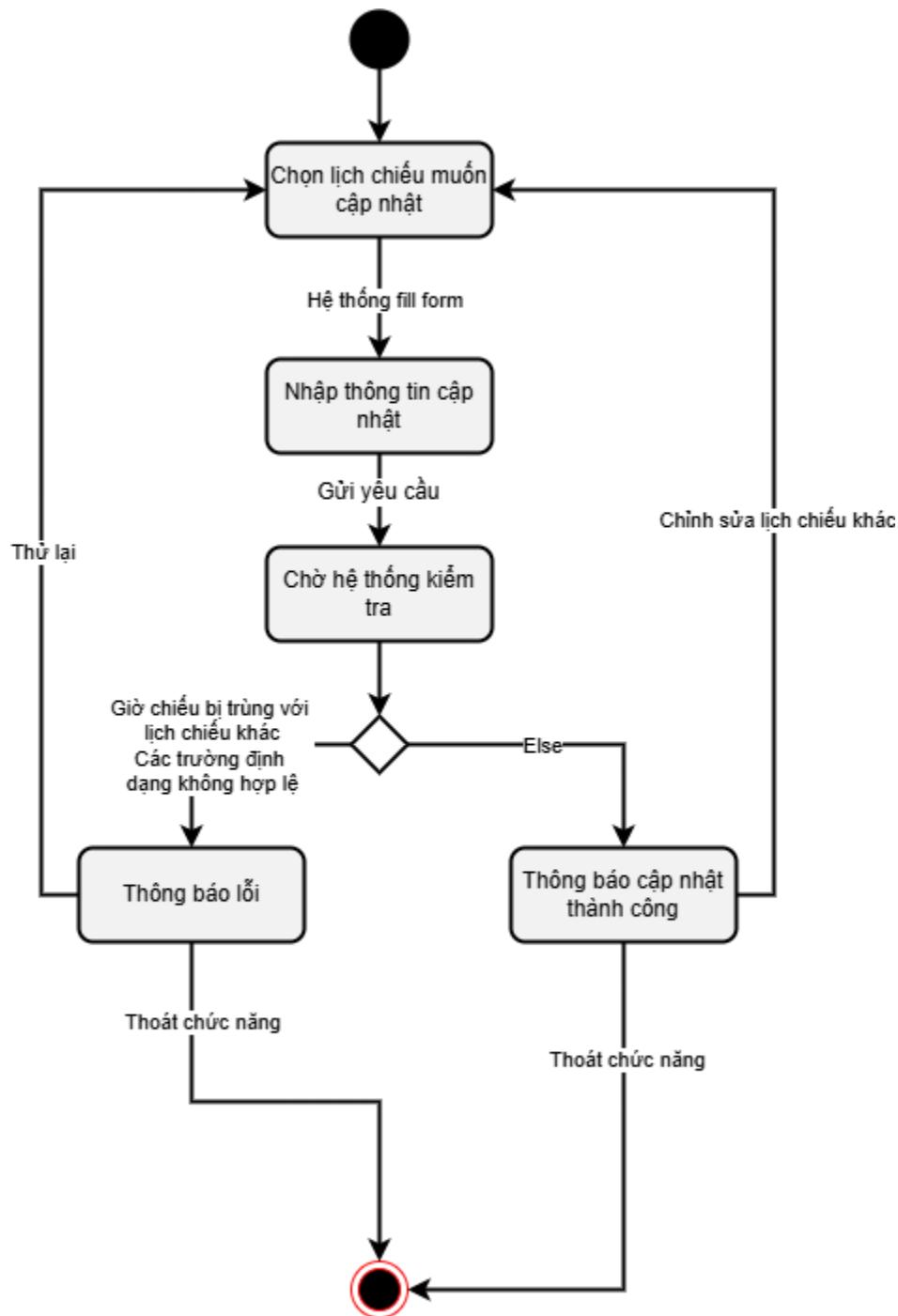


Figure 22 Cập nhật lịch chiếu

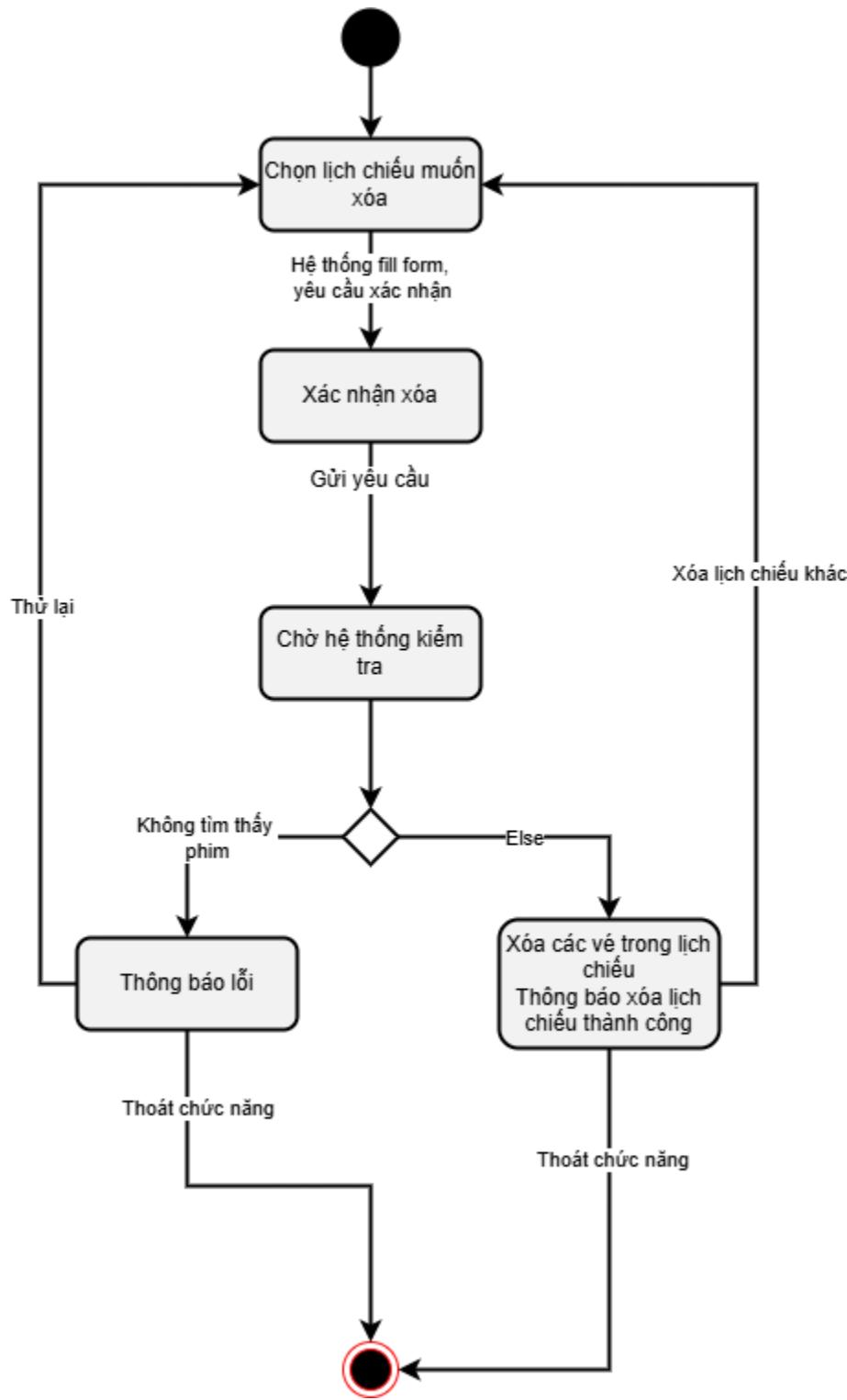


Figure 23 Xóa lịch chiếu

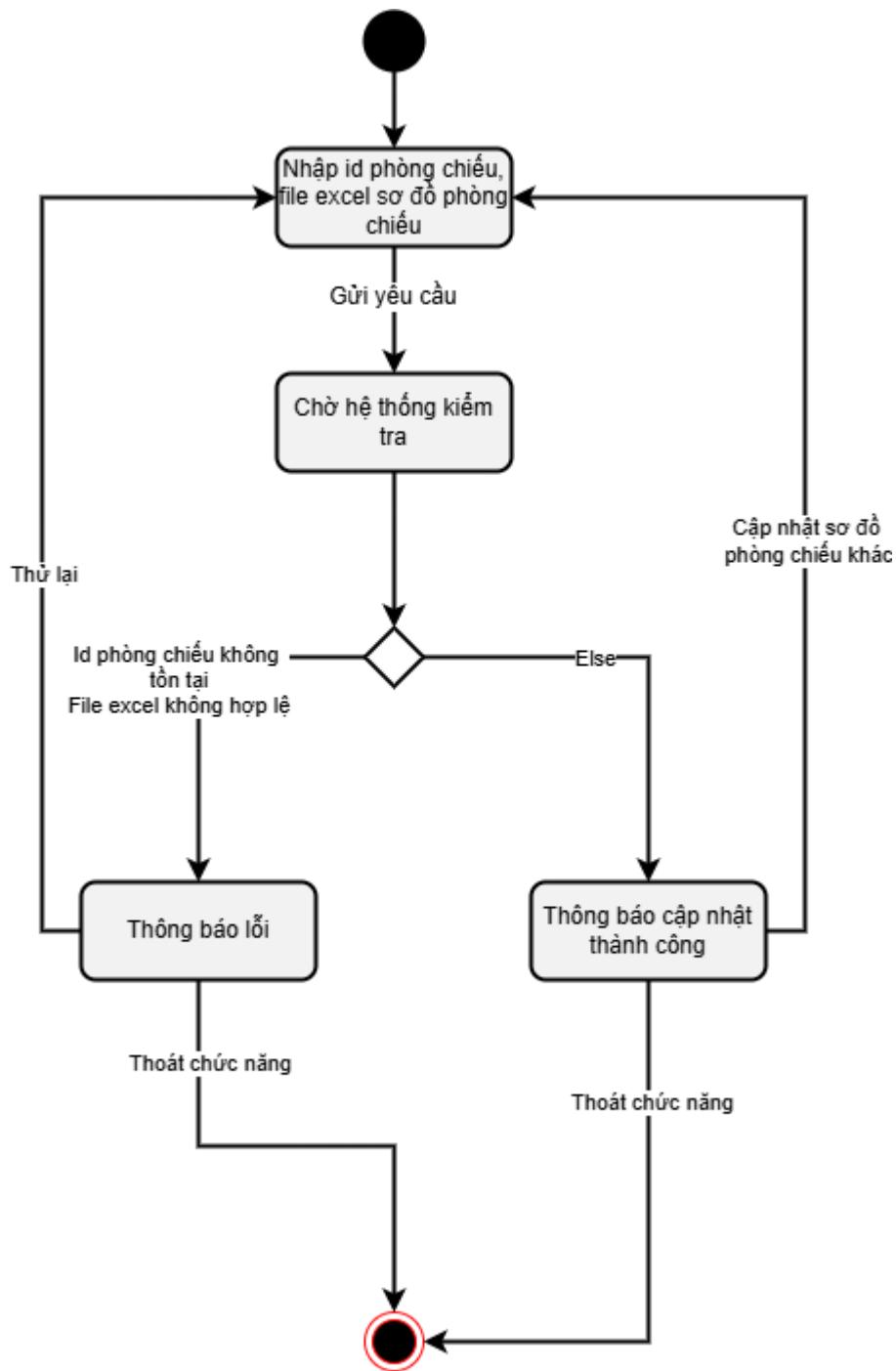


Figure 24 Cập nhật sơ đồ phòng chiếu

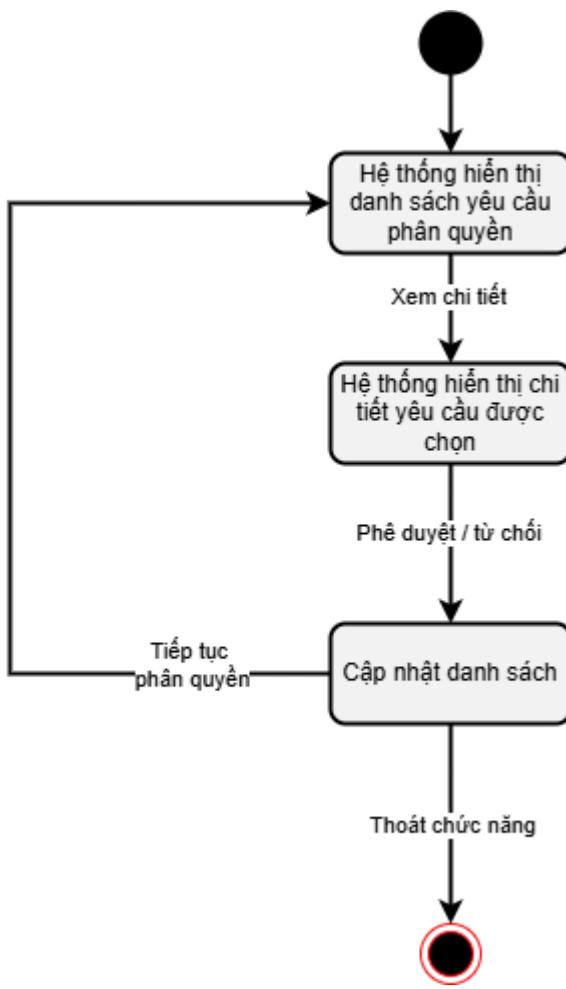


Figure 25 Phân quyền

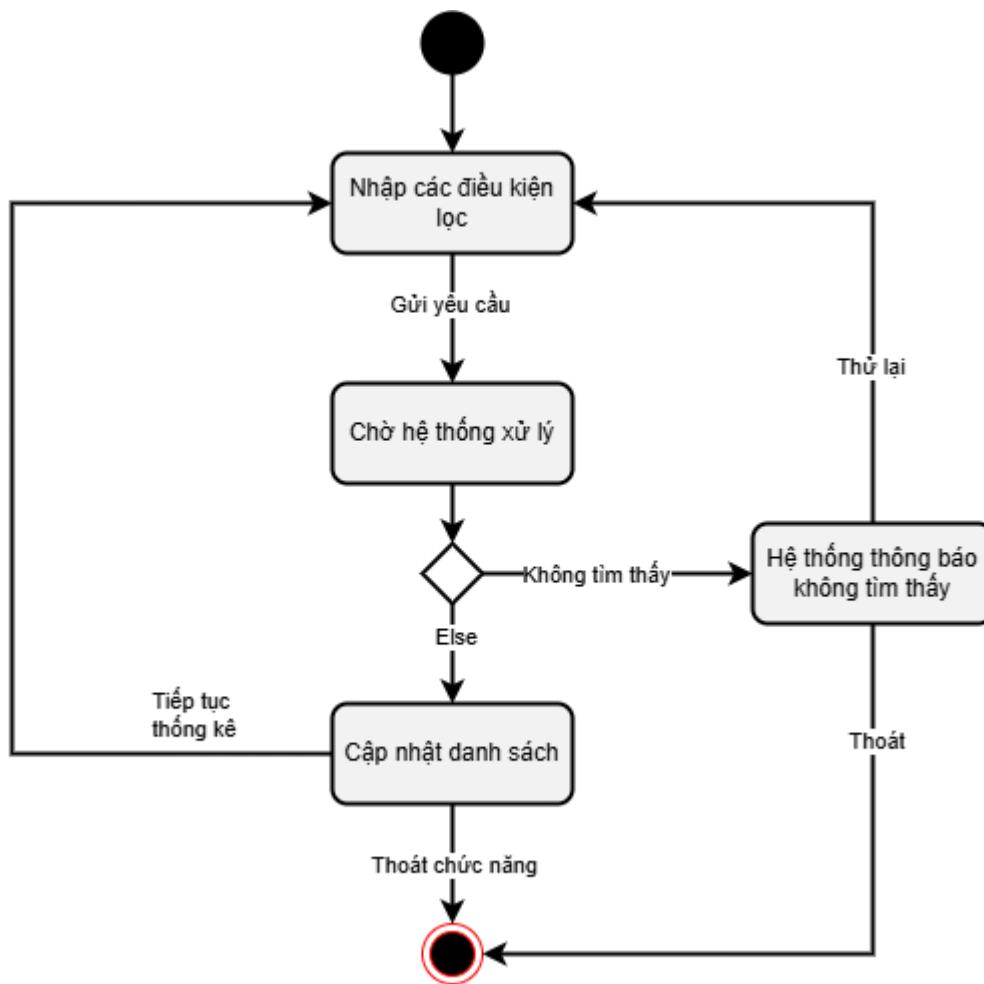


Figure 26 Thông kê

2.2 Xác định yêu cầu chức năng

2.2.1 Xác định các yêu cầu mức cao

1. Quản lý phim

- Thêm/Xóa/Cập nhật phim: Hệ thống cho phép quản trị viên thêm phim mới, xóa phim hoặc cập nhật thông tin về phim như tên phim, đạo diễn, diễn viên, thể loại, thời lượng, và mô tả.
- Quản lý lịch chiếu: Sắp xếp lịch chiếu cho các phim, bao gồm ngày giờ chiếu và phòng chiếu.

2. Quản lý lịch chiếu

- Lập lịch chiếu phim: Xác định các suất chiếu cụ thể cho từng phim trong ngày, tuần, hoặc tháng.

- Cập nhật lịch chiếu: Cho phép quản trị viên cập nhật hoặc thay đổi lịch chiếu theo nhu cầu.
3. Quản lý phòng chiếu
- Thêm/Xóa/Cập nhật phòng chiếu: Quản lý thông tin về các phòng chiếu như số lượng ghế, cấu hình ghế (VIP, thường), và các thiết bị trong phòng chiếu.
 - Quản lý tình trạng phòng chiếu: Kiểm tra và cập nhật tình trạng hoạt động của các phòng chiếu.
4. Quản lý vé
- Đặt vé: Khách hàng có thể chọn phim, suất chiếu, và ghế để đặt vé trực tuyến hoặc tại quầy.
 - Thanh toán vé: Cung cấp nhiều phương thức thanh toán cho khách hàng như thẻ tín dụng, ví điện tử, và tiền mặt.
5. Quản lý khách hàng
- Đăng ký/Đăng nhập: Khách hàng có thể tạo tài khoản, đăng nhập, và quản lý thông tin cá nhân.
 - Quản lý thông tin cá nhân: Khách hàng có thể cập nhật thông tin cá nhân, xem lịch sử đặt vé, và quản lý điểm thưởng.
6. Quản lý nhân viên
- Phân quyền và quản lý nhân viên: Quản trị viên có thể thêm, xóa, và phân quyền cho nhân viên.
7. Báo cáo và phân tích
- Báo cáo doanh thu: Hệ thống cung cấp các báo cáo về doanh thu theo ngày, tuần, tháng, hoặc theo phim.
 - Phân tích khách hàng: Thống kê và phân tích hành vi khách hàng để tối ưu hóa chiến lược kinh doanh.

2.2.2 Xác định các Usecase

U1. Xem thông tin phim	U13. Quản lý rạp
U2. Đặt vé	U14. Quản lý lịch chiếu
U3. Đặt đồ ăn	U15. Quản lý phòng chiếu
U4. Quản lý giá vé phim	U16. Phân quyền
U5. In vé	U17. Thống kê doanh thu
U6. Đăng nhập	U18. Thống kê phim
U7. Đăng ký	U19. Thống kê rạp
U8. Cập nhật tài khoản	U20. Thống kê đồ ăn
U9. Xem hóa đơn	U21. Thống kê nhân viên
U10. In mã QR	U22. Xem thông tin khuyến mãi
U11. Cập nhật sơ đồ phòng chiếu	U23. Quản lý khuyến mãi
U12. Quản lý phim	U24. Quản lý đồ ăn

2.2.3 Khảo sát usecase

- ❖ Khi truy cập vào ứng dụng, người dùng có thể xem thông tin phim (U1), xem thông tin khuyến mãi (U22)
- ❖ Người dùng cần đăng nhập vào hệ thống (U6) để có thể sử dụng chức năng đặt vé (U2), đặt đồ ăn (U3), xem hóa đơn (U9), cập nhật tài khoản (U8). Người dùng chưa có tài khoản có thể đăng ký (U7)
- ❖ Người dùng có thể xem các thông tin cụ thể của phim (U1) bao gồm ngày khởi chiếu, giới thiệu phim, diễn viên trong phim, thể loại
- ❖ Sau khi người dùng xem thông tin phim (U1), người dùng có thể tiến hành đặt vé (U2). Trong quy trình đặt vé, người dùng sẽ chọn rạp, lịch chiếu phù hợp với mình sau đó hệ thống sẽ hiển thị sơ đồ phòng chiếu cho người dùng chọn chỗ ngồi, người dùng sau khi chọn được chỗ ngồi ưng ý sẽ có thể tiến hành thanh toán. Hệ thống sẽ có nhiệm vụ thông báo thông tin đặt vé cho người dùng, gửi email xác nhận, lưu trữ hóa đơn lên hệ thống. Người dùng sau đó có thể xem lại hóa đơn của mình (U9)
- ❖ Nhân viên khi truy cập vào ứng dụng dành cho nhân viên sẽ phải đăng nhập (U6), sau đó nhân viên có thể cập nhật tài khoản (U8). Nhân viên có thể đặt vé (U2) trực tiếp giúp khách hàng qua quy trình tương tự như quy trình của khách hàng, ngoại trừ bước thanh toán nhân viên sẽ tạo 1 QR code (U10) để người dùng có thể thanh toán.
- ❖ Quản trị viên cần đăng nhập (U6) để sử dụng app quản trị. Sau khi đăng nhập, quản trị viên có thể cập nhật thông tin tài khoản (U8), sử dụng các chức năng của quản trị viên: phân quyền (U16), quản lý danh mục (U11->U15, U23->U24), thống kê (U17 -> U21). Ban đầu khi hệ thống vừa được tạo sẽ có sẵn 1 tài khoản quản trị.

2.2.4 Sắp xếp ưu tiên các usecase

Xanh

- | | |
|---------------------------------|--------------------------|
| U1. Xem thông tin phim | U12. Quản lý phim |
| U2. Đặt vé | U13. Quản lý rạp |
| U3. Đặt đồ ăn | U14. Quản lý lịch chiếu |
| U4. Quản lý giá vé phim | U15. Quản lý phòng chiếu |
| U6. Đăng nhập | U18. Thống kê phim |
| U7. Đăng ký | U19. Thống kê rạp |
| U9. Xem hóa đơn | U20. Thống kê đồ ăn |
| U11. Cập nhật sơ đồ phòng chiếu | |

Vàng

- | | |
|-------------------------|-------------------------|
| U8. Cập nhật tài khoản | U21. Thống kê nhân viên |
| U10. In mã QR | U23. Quản lý khuyến mãi |
| U16. Phân quyền | U24. Quản lý đồ ăn |
| U17. Thống kê doanh thu | |

Đỏ

- | | |
|-----------|-------------------------------|
| U5. In vé | U22. Xem thông tin khuyến mãi |
|-----------|-------------------------------|

2.2.5 Biểu đồ usecase

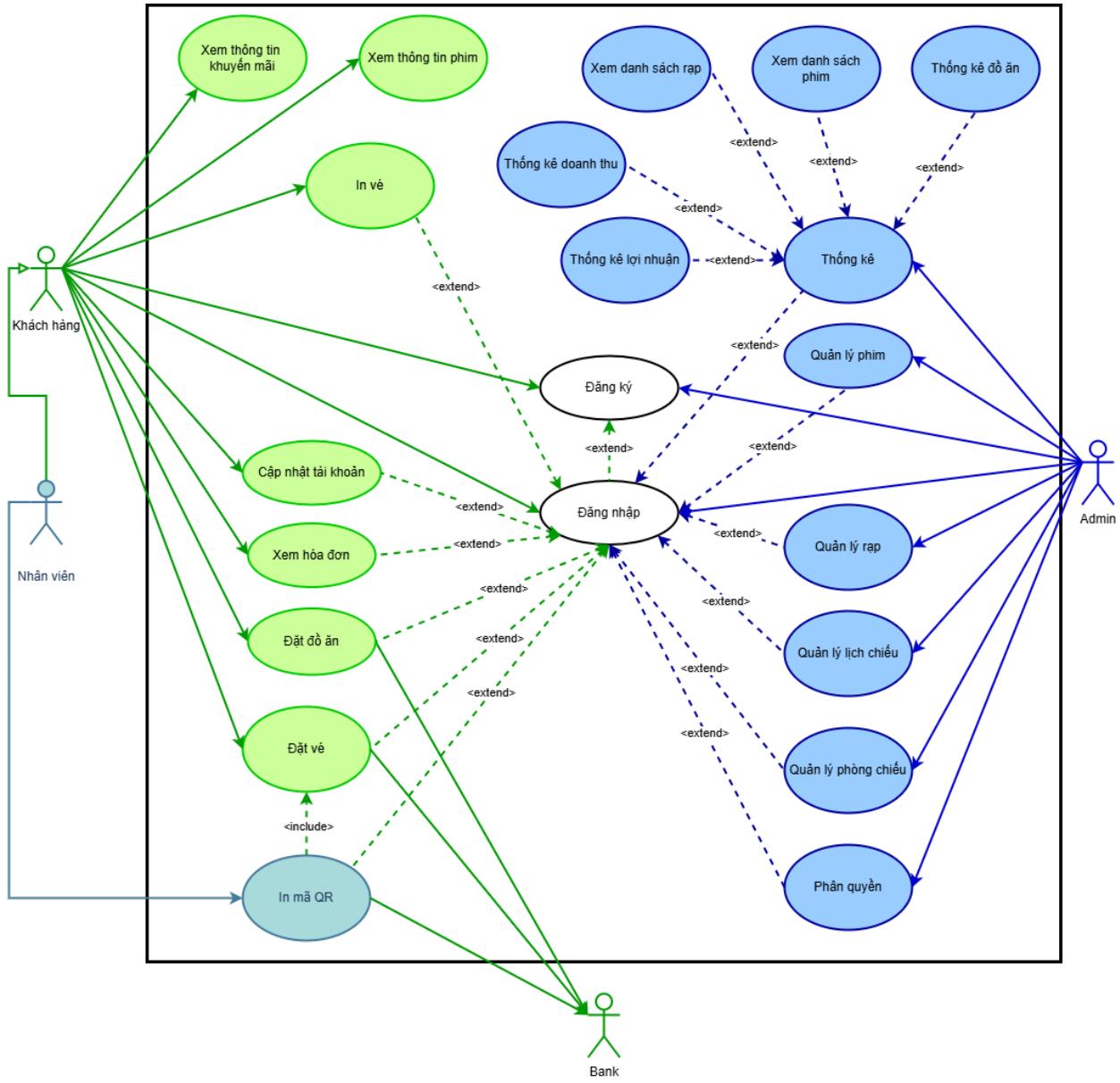


Figure 27 Usecase tổng quan

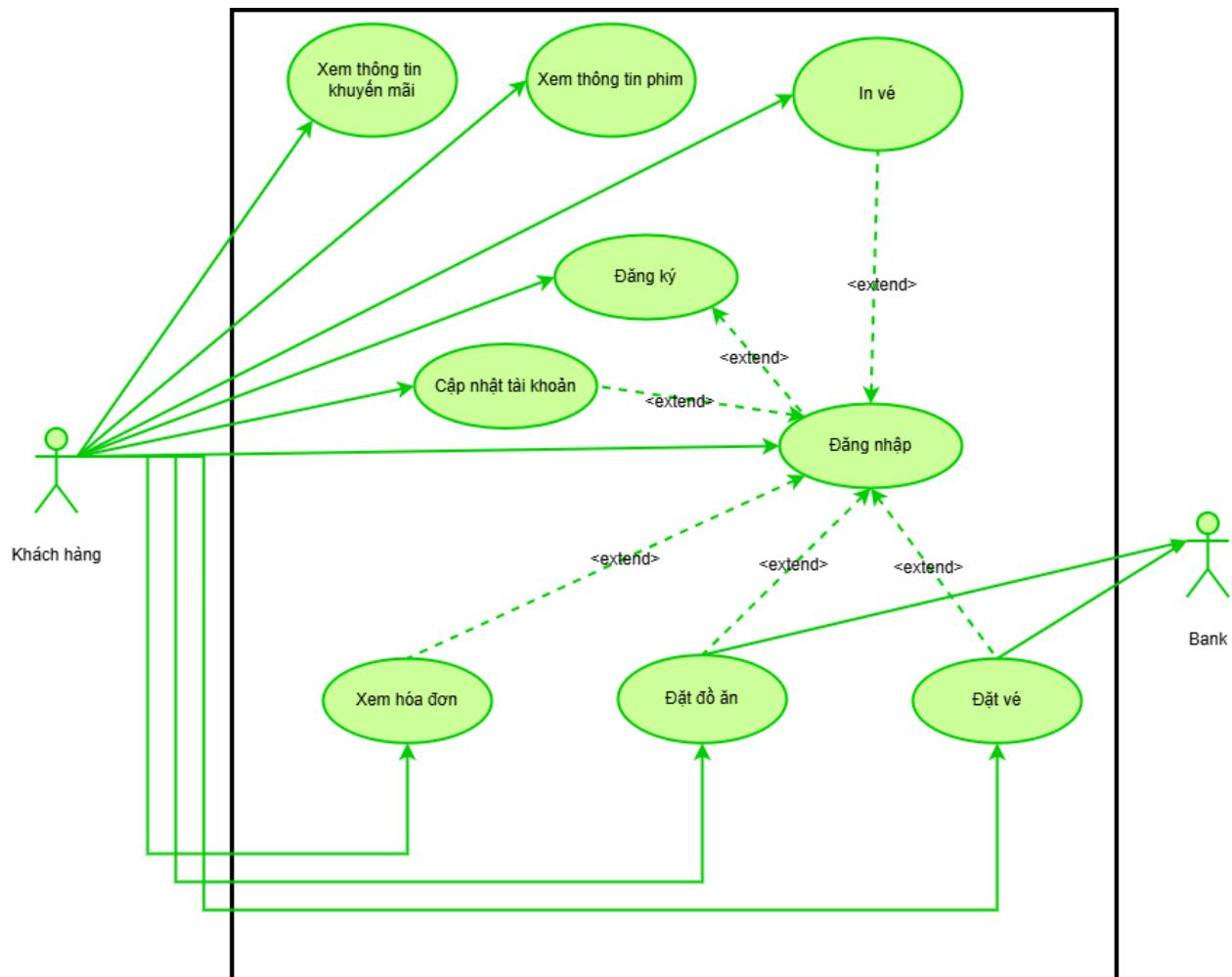


Figure 28 Usecase khách hàng

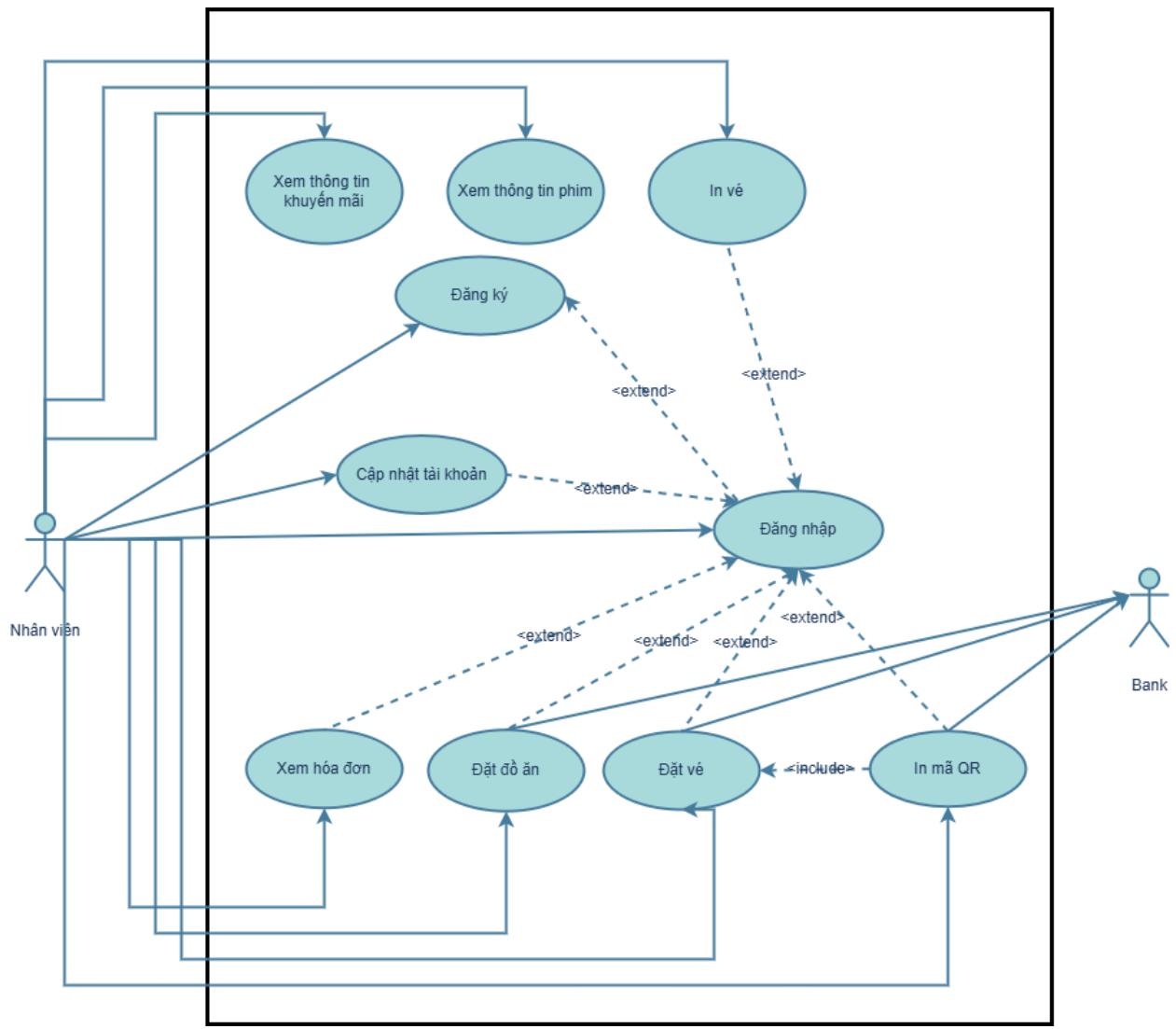


Figure 29 Usecase nhân viên

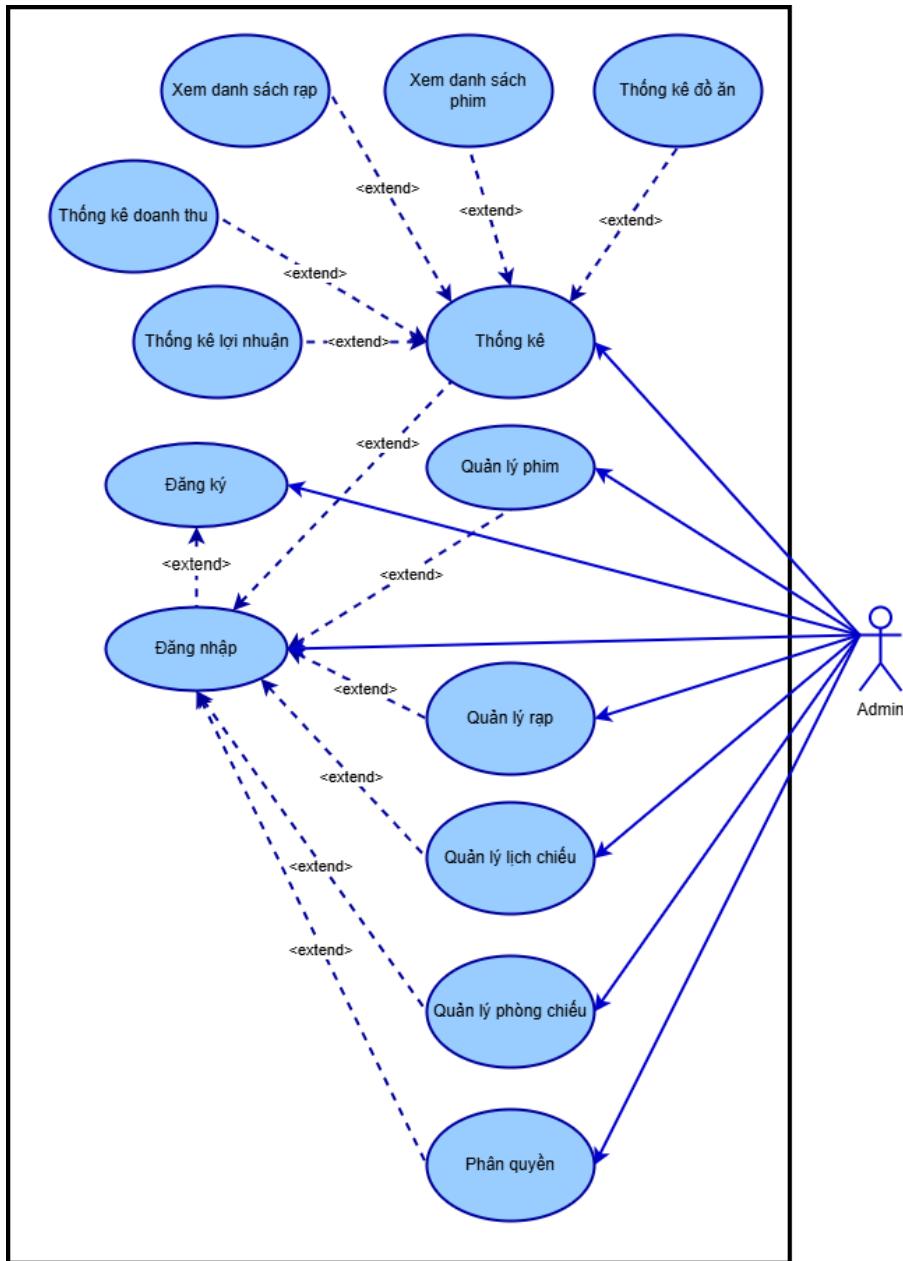


Figure 30 Usecase admin

2.2.6 Đặc tả usecase

Tên usecase	Đăng nhập
Tác nhân chính	Người dùng (khách hàng, nhân viên, quản lý)
Điều kiện trước	Người dùng đã có tài khoản trên hệ thống
Đảm bảo tối thiểu	Hệ thống cho phép người dùng đăng nhập lại
Điều kiện sau	Người dùng đăng nhập thành công
Chuỗi sự kiện chính	
1. Người dùng chọn chức năng đăng nhập trên giao diện chính của hệ thống 2. Hệ thống hiển thị form đăng nhập 3. Người dùng nhập tài khoản và mật khẩu của mình 4. Hệ thống kiểm tra tính hợp lệ của tài khoản và mật khẩu 5. Hệ thống hiển thị giao diện chính tương ứng với tác nhân	
Ngoại lệ	
4.1 Người dùng nhập tài khoản và mật khẩu sai 4.1.1 Hệ thống thông báo lỗi và yêu cầu nhập lại 4.2 Tài khoản người dùng đăng nhập không tồn tại 4.2.1 Hệ thống thông báo lỗi và yêu cầu người sử dụng đăng ký	

Tên usecase	Đăng ký
Tác nhân chính	Người dùng (khách hàng)
Điều kiện trước	
Đảm bảo tối thiểu	Hệ thống cho phép người dùng đăng ký nhiều lần
Điều kiện sau	Người dùng đăng ký thành công

Chuỗi sự kiện chính
<ol style="list-style-type: none"> 1. Người dùng chọn chức năng đăng ký trên giao diện chính của hệ thống 2. Hệ thống hiển thị form đăng ký 3. Người dùng nhập tài khoản mật khẩu và các thông tin cơ bản 4. Hệ thống kiểm tra tính hợp lệ của tài khoản và mật khẩu 5. Hệ thống hiển thị giao diện đăng nhập

Ngoại lệ
<ol style="list-style-type: none"> 4.1 Người dùng nhập tài khoản đã tồn tại <ol style="list-style-type: none"> 4.1.1 Hệ thống thông báo lỗi và yêu cầu nhập lại 4.2 Người dùng nhập xác nhận mật khẩu sai <ol style="list-style-type: none"> 4.2.1 Hệ thống thông báo lỗi và yêu cầu kiểm tra lại 4.3 Email không hợp lệ <ol style="list-style-type: none"> 4.3.1 Hệ thống thông báo lỗi và yêu cầu nhập lại

Tên usecase	Cập nhật thông tin
Tác nhân chính	Người dùng (khách hàng)
Điều kiện trước	Người dùng đã đăng nhập
Đảm bảo tối thiểu	Hệ thống cho phép người dùng đổi thông tin nhiều lần
Điều kiện sau	Người dùng cập nhật thông tin thành công
Chuỗi sự kiện chính	
<ol style="list-style-type: none"> 1. Người dùng chọn chức năng cập nhật thông tin trên giao diện chính của hệ thống 2. Hệ thống hiển thị form thông tin với những thông tin hiện tại 3. Người dùng nhập các thông tin muốn thay đổi (ngoại trừ email) 4. Hệ thống cập nhật thông tin lên cloud 5. Hệ thống hiển thị thông báo thành công 	
Ngoại lệ	
<p>6 Người dùng muốn đổi mật khẩu</p> <p>6.1 Hệ thống yêu cầu nhập mật khẩu cũ, và nhập 2 lần mật khẩu mới để xác nhận</p> <p>6.2 Hệ thống kiểm tra tính hợp lệ</p> <ul style="list-style-type: none"> 6.2.1 Người dùng nhập xác nhận mật khẩu cũ sai => Hệ thống thông báo lỗi và yêu cầu kiểm tra lại 6.2.2 Người dùng nhập xác nhận mật khẩu sai => Hệ thống thông báo lỗi và yêu cầu kiểm tra lại 6.2.3 Thông tin hợp lệ => Hệ thống cập nhật lên cloud, hiển thị thông báo đổi mật khẩu thành công 	

Tên usecase	Xem hóa đơn
Tác nhân chính	Người dùng (khách hàng)
Điều kiện trước	Người dùng đã đăng nhập
Đảm bảo tối thiểu	Hệ thống cho phép người dùng xem hóa đơn nhiều lần
Điều kiện sau	Hệ thống hiển thị thông tin hóa đơn cho người dùng
Chuỗi sự kiện chính	<ol style="list-style-type: none"> 1. Người dùng chọn chức năng xem các hóa đơn đã mua 2. Hệ thống truy xuất từ cloud và trả về danh sách các hóa đơn đã mua của khách hàng, mỗi hóa đơn ở đây bao gồm ngày mua, giá tiền, loại hóa đơn (vé xem phim hay đồ ăn)
Ngoại lệ	<ol style="list-style-type: none"> 1. Người dùng nhấn vào 1 hóa đơn cụ thể 2. Hệ thống hiển thị chi tiết hóa đơn cho người dùng (sản phẩm, số lượng sản phẩm, rạp chiếu,...)

Tên usecase	In vé
Tác nhân chính	Người dùng (khách hàng)
Điều kiện trước	Người dùng đã đăng nhập
Đảm bảo tối thiểu	Hệ thống cho phép người dùng thử lại nhiều lần
Điều kiện sau	Hệ thống in vé cho người dùng
Chuỗi sự kiện chính	
1. Người dùng chọn chức năng in vé 2. Hệ thống bật máy quét, yêu cầu người dùng cung cấp mã vạch 3. Người dùng cung cấp mã vạch cho máy quét 4. Máy in vé cho người dùng	
Ngoại lệ	
3.1 Mã đã sử dụng => Hệ thống báo lỗi cho người dùng, yêu cầu thử lại hoặc kết thúc phiên 3.2 Người dùng không cung cấp mã vạch => Sau 1 phút hệ thống tự kết thúc phiên 3.3 Mã không hợp lệ => Hệ thống báo lỗi cho người dùng, yêu cầu thử lại hoặc kết thúc phiên	

Tên usecase	Đặt vé
Tác nhân chính	Người dùng (khách hàng)
Điều kiện trước	Người dùng đã đăng nhập
Đảm bảo tối thiểu	Hệ thống cho phép người dùng thử lại nhiều lần
Điều kiện sau	Người dùng đặt vé thành công
Chuỗi sự kiện chính	
1. Người dùng click đặt vé 2. Hệ thống hiển thị rạp chiếu và suất chiếu tại rạp trong các ngày chiếu 3. Người dùng chọn suất chiếu, rạp chiếu, ngày chiếu 4. Hệ thống hiển thị sơ đồ phòng chiếu, các ghế còn trống, các ghế đã đặt cho người dùng 5. Người dùng chọn 1 hoặc nhiều ghế muốn đặt 6. Hệ thống cập nhật giá xem trước cho người dùng 7. Người dùng click vào thanh toán 8. Hệ thống hiển thị trang thanh toán 9. Người dùng chọn phương thức thanh toán, nhập thông tin thanh toán và tiến hành thanh toán 10. Hệ thống kiểm tra hợp lệ, thông báo đặt vé thành công 11. Hệ thống gửi email cho người dùng	
Ngoại lệ	
8.1 Người dùng nhập voucher 8.1.1 Nếu hợp lệ, hệ thống thông báo áp dụng voucher thành công và hiển thị giá sau áp dụng 8.1.2 Nếu không hợp lệ, hệ thống thông báo lỗi, yêu cầu thử lại 8.2 Người dùng yêu cầu thoát => Quá trình đặt vé bị hủy	

Tên usecase	Đặt đồ ăn
Tác nhân chính	Người dùng (khách hàng)
Điều kiện trước	Người dùng đã đăng nhập
Đảm bảo tối thiểu	Hệ thống cho phép người dùng thử lại nhiều lần
Điều kiện sau	Người dùng đặt đồ ăn thành công
Chuỗi sự kiện chính	
1. Người dùng click đặt đồ ăn 2. Hệ thống hiển thị danh sách các món ăn tại các rạp 3. Người dùng chọn rạp, chọn món ăn, số lượng 4. Hệ thống cập nhật giá xem trước cho người dùng 5. Người dùng click vào thanh toán 6. Hệ thống hiển thị trang thanh toán 7. Người dùng chọn phương thức thanh toán, nhập thông tin thanh toán và tiến hành thanh toán 8. Hệ thống kiểm tra hợp lệ, thông báo đặt vé thành công 9. Hệ thống gửi email cho người dùng	
Ngoại lệ	
7.1 Người dùng nhập voucher 7.1.1 Nếu hợp lệ, hệ thống thông báo áp dụng voucher thành công và hiển thị giá sau áp dụng 7.1.2 Nếu không hợp lệ, hệ thống thông báo lỗi, yêu cầu thử lại 7.2 Người dùng yêu cầu thoát => Quá trình đặt bị hủy	

Tên usecase	Quản lý phim
Tác nhân chính	Admin
Điều kiện trước	Admin đã đăng nhập
Đảm bảo tối thiểu	Hệ thống cho phép Admin tra cứu thông tin phim và click vào dòng để tự fill thông tin lên form
Điều kiện sau	Admin hoàn thành quản lý phim
Chuỗi sự kiện chính	
A. Thêm phim <ol style="list-style-type: none"> 1 Admin nhập thông tin phim muốn thêm 2 Hệ thống kiểm tra các ràng buộc tối thiểu(not null, isNumber.....) 3 Hệ thống kiểm tra trùng lặp 4 Phim được thêm vào cloud 	
B. Cập nhật phim <ol style="list-style-type: none"> 1 Admin nhập thông tin phim sau chỉnh sửa (ngoại trừ id không được sửa) 2 Hệ thống kiểm tra các ràng buộc tối thiểu(not null, isNumber.....) 3 Phim được cập nhật trên cloud 	
C. Xóa phim <ol style="list-style-type: none"> 1 Admin nhập id phim muốn xóa 2 Hệ thống kiểm tra tồn tại 3 Hệ thống yêu cầu người dùng xác nhận 4 Phim được xóa khỏi cloud 	
Ngoại lệ	
Nếu trong quá trình thêm/sửa/xóa mà thông tin đầu vào bị lỗi, hệ thống báo lỗi và yêu cầu người dùng thử lại	

Tên usecase	Quản lý rạp
Tác nhân chính	Admin
Điều kiện trước	Admin đã đăng nhập
Đảm bảo tối thiểu	Hệ thống cho phép Admin tra cứu thông tin rạp và click vào dòng để tự fill thông tin lên form
Điều kiện sau	Admin hoàn thành quản lý rạp
Chuỗi sự kiện chính	
A. Thêm rạp <ol style="list-style-type: none"> 1 Admin nhập thông tin rạp muốn thêm 2 Hệ thống kiểm tra các ràng buộc tối thiểu(not null, isNumber.....) 3 Hệ thống kiểm tra trùng lặp 4 Phim được thêm vào cloud 	
B. Cập nhật phim <ol style="list-style-type: none"> 1 Admin nhập thông tin rạp sau chỉnh sửa (ngoại trừ id không được sửa) 2 Hệ thống kiểm tra các ràng buộc tối thiểu(not null, isNumber.....) 3 Rạp được cập nhật trên cloud 	
C. Xóa phim <ol style="list-style-type: none"> 1 Admin nhập id rạp muốn xóa 2 Hệ thống kiểm tra tồn tại 3 Hệ thống yêu cầu người dùng xác nhận 4 Rạp được xóa khỏi cloud 	
Ngoại lệ	
Nếu trong quá trình thêm/sửa/xóa mà thông tin đầu vào bị lỗi, hệ thống báo lỗi và yêu cầu người dùng thử lại	

Tên usecase	Quản lý phòng chiếu
Tác nhân chính	Admin
Điều kiện trước	Admin đã đăng nhập
Đảm bảo tối thiểu	Hệ thống cho phép Admin tra cứu thông tin phòng chiếu và click vào dòng để tự fill thông tin lên form
Điều kiện sau	Admin hoàn thành quản lý phòng chiếu
Chuỗi sự kiện chính	
A. Thêm phòng chiếu <ol style="list-style-type: none"> 1 Admin nhập thông tin phòng chiếu muốn thêm (bao gồm file excel sơ đồ phòng chiếu) 2 Hệ thống kiểm tra các ràng buộc tối thiểu(not null, isNumber.....) 3 Hệ thống kiểm tra trùng lặp 4 Phòng chiếu được thêm vào cloud 	
B. Cập nhật phòng chiếu <ol style="list-style-type: none"> 1 Admin nhập thông tin phòng chiếu sau chỉnh sửa (ngoại trừ id phòng chiếu không được sửa) 2 Hệ thống kiểm tra các ràng buộc tối thiểu(not null, isNumber.....) 3 Phòng chiếu được cập nhật trên cloud 	
C. Xóa phòng chiếu <ol style="list-style-type: none"> 1 Admin nhập id phòng chiếu muốn xóa 2 Hệ thống kiểm tra tồn tại 3 Hệ thống yêu cầu người dùng xác nhận 4 Phòng chiếu được xóa khỏi cloud 	
Ngoại lệ	
Nếu trong quá trình thêm/sửa/xóa mà thông tin đầu vào bị lỗi, hệ thống báo lỗi và yêu cầu người dùng thử lại	

Tên usecase	Cập nhật sơ đồ phòng chiếu
Tác nhân chính	Admin
Điều kiện trước	Admin đã đăng nhập
Đảm bảo tối thiểu	Hệ thống cho phép Admin cập nhật thông tin sơ đồ phòng chiếu và bằng file excel sơ đồ phòng chiếu
Điều kiện sau	Admin hoàn thành quản lý sơ đồ phòng chiếu
Chuỗi sự kiện chính	
1 Admin nhập thông tin sơ đồ phòng chiếu sau chỉnh sửa (file excel, thông tin phòng chiếu nhập form)	
2 Hệ thống kiểm tra các ràng buộc tối thiểu(not null, isNumber.....)	
3 Sơ đồ phòng chiếu được cập nhật trên cloud	
Ngoại lệ	
Nếu trong quá trình cập nhật mà thông tin đầu vào bị lỗi, hệ thống báo lỗi và yêu cầu người dùng thử lại	

Tên usecase	Quản lý đồ ăn
Tác nhân chính	Admin
Điều kiện trước	Admin đã đăng nhập
Đảm bảo tối thiểu	Hệ thống cho phép Admin tra cứu thông tin đồ ăn và click vào dòng để tự fill thông tin lên form
Điều kiện sau	Admin hoàn thành quản lý đồ ăn
Chuỗi sự kiện chính	
A. Thêm đồ ăn <ul style="list-style-type: none"> 1 Admin nhập thông tin đồ ăn muốn thêm 2 Hệ thống kiểm tra các ràng buộc tối thiểu(not null, isNumber.....) 3 Hệ thống kiểm tra trùng lặp 4 Đồ ăn được thêm vào cloud 	
B. Cập nhật đồ ăn <ul style="list-style-type: none"> 1 Admin nhập thông tin đồ ăn sau chỉnh sửa (ngoại trừ id không được sửa) 2 Hệ thống kiểm tra các ràng buộc tối thiểu(not null, isNumber.....) 3 Đồ ăn được cập nhật trên cloud 	
C. Xóa đồ ăn <ul style="list-style-type: none"> 1 Admin nhập id đồ ăn muốn xóa 2 Hệ thống kiểm tra tồn tại 3 Hệ thống yêu cầu người dùng xác nhận 4 Đồ ăn được xóa khỏi cloud 	
Ngoại lệ	
Nếu trong quá trình thêm/sửa/xóa mà thông tin đầu vào bị lỗi, hệ thống báo lỗi và yêu cầu người dùng thử lại	

Tên usecase	Quản lý lịch chiếu
Tác nhân chính	Admin
Điều kiện trước	Admin đã đăng nhập
Đảm bảo tối thiểu	Hệ thống cho phép Admin tra cứu thông tin lịch chiếu và click vào dòng để tự fill thông tin lên form
Điều kiện sau	Admin hoàn thành quản lý đồ ăn
Chuỗi sự kiện chính	
A. Thêm đồ ăn	
1 Admin nhập thông tin lịch chiếu muốn thêm	
2 Hệ thống kiểm tra các ràng buộc tối thiểu(not null, isNumber.....)	
3 Hệ thống kiểm tra overlap, trùng lặp	
4 Lịch chiếu được thêm vào cloud	
B. Cập nhật lịch chiếu	
1 Admin nhập thông tin lịch chiếu sau chỉnh sửa (ngoại trừ id không được sửa)	
2 Hệ thống kiểm tra các ràng buộc tối thiểu(not null, isNumber.....)	
3 Hệ thống kiểm tra overlap, trùng lặp	
4 Lịch chiếu được cập nhật trên cloud	
C. Xóa lịch chiếu	
1 Admin nhập id lịch chiếu muốn xóa	
2 Hệ thống kiểm tra tồn tại	
3 Hệ thống yêu cầu người dùng xác nhận	
4 Lịch chiếu được xóa khỏi cloud	
Ngoại lệ	
Nếu trong quá trình thêm/sửa/xóa mà thông tin đầu vào bị lỗi, hệ thống báo lỗi và yêu cầu người dùng thử lại	

Tên usecase	Quản lý giá vé
Tác nhân chính	Admin
Điều kiện trước	Admin đã đăng nhập
Đảm bảo tối thiểu	Hệ thống cho phép Admin tra cứu thông tin giá vé và click vào dòng để tự fill thông tin lên form
Điều kiện sau	Admin hoàn thành quản lý giá vé
Chuỗi sự kiện chính	
A. Thêm giá vé <ol style="list-style-type: none"> 1 Admin nhập thông tin giá vé muốn thêm 2 Hệ thống kiểm tra các ràng buộc tối thiểu(not null, isNumber.....) 3 Hệ thống kiểm tra overlap, trùng lặp 4 Giá vé được thêm vào cloud 	
B. Cập nhật giá vé <ol style="list-style-type: none"> 1 Admin nhập thông tin giá vé sau chỉnh sửa 2 Hệ thống kiểm tra các ràng buộc tối thiểu(not null, isNumber.....) 3 Hệ thống kiểm tra overlap, trùng lặp 4 Giá vé được cập nhật trên cloud 	
Ngoại lệ	
Nếu trong quá trình thêm/sửa mà thông tin đầu vào bị lỗi, hệ thống báo lỗi và yêu cầu người dùng thử lại	

Tên usecase	Thống kê phim
Tác nhân chính	Admin
Điều kiện trước	Admin đã đăng nhập
Đảm bảo tối thiểu	Hệ thống cho phép Admin thống kê phim theo filter
Điều kiện sau	Admin hoàn thành thống kê phim
Chuỗi sự kiện chính	
1 Admin nhập các điều kiện lọc (tên phim, ngày chiếu....)	
2 Hệ thống lấy dữ liệu về và lọc theo bộ lọc	
3 Hệ thống hiển thị kết quả	
Ngoại lệ	
<ul style="list-style-type: none"> ➡ Nếu người dùng không nhập bộ lọc, hiển thị toàn bộ danh sách phim ➡ Nếu kết quả trả về là rỗng, hệ thống sẽ báo lỗi, yêu cầu người dùng kiểm tra lại 	

Tên usecase	Thống kê rạp
Tác nhân chính	Admin
Điều kiện trước	Admin đã đăng nhập
Đảm bảo tối thiểu	Hệ thống cho phép Admin thống kê rạp theo filter
Điều kiện sau	Admin hoàn thành thống kê rạp
Chuỗi sự kiện chính	
1 Admin nhập các điều kiện lọc (tên rạp, ngày thêm....)	
2 Hệ thống lấy dữ liệu về và lọc theo bộ lọc	
3 Hệ thống hiển thị kết quả	
Ngoại lệ	
<ul style="list-style-type: none"> ➡ Nếu người dùng không nhập bộ lọc, hiển thị toàn bộ danh sách rạp ➡ Nếu kết quả trả về là rỗng, hệ thống sẽ báo lỗi, yêu cầu người dùng kiểm tra lại 	

Tên usecase	Thống kê đồ ăn
Tác nhân chính	Admin
Điều kiện trước	Admin đã đăng nhập
Đảm bảo tối thiểu	Hệ thống cho phép Admin thống kê rạp theo filter
Điều kiện sau	Admin hoàn thành thống kê rạp
Chuỗi sự kiện chính	
1 Admin nhập các điều kiện lọc (tên món ăn, giá....)	
2 Hệ thống lấy dữ liệu về và lọc theo bộ lọc	
3 Hệ thống hiển thị kết quả	
Ngoại lệ	
<ul style="list-style-type: none"> ➡ Nếu người dùng không nhập bộ lọc, hiển thị toàn bộ danh sách đồ ăn ➡ Nếu kết quả trả về là rỗng, hệ thống sẽ báo lỗi, yêu cầu người dùng kiểm tra lại 	

Tên usecase	Phân quyền
Tác nhân chính	Admin
Điều kiện trước	Admin đã đăng nhập
Đảm bảo tối thiểu	Hệ thống cho phép xem danh sách các yêu cầu đăng ký
Điều kiện sau	Admin hoàn thành phê duyệt phân quyền
Chuỗi sự kiện chính	
1 Hệ thống hiển thị các yêu cầu đăng ký (họ tên, id, ngày sinh, email, vị trí đăng ký...)	
2 Admin chọn phê duyệt hoặc từ chối	
3 Yêu cầu đăng ký được xử lý, cập nhật lên cloud	
Ngoại lệ	
 Nếu trong quá trình cập nhật lên cloud bị lỗi timeout, yêu cầu được giữ nguyên không cập nhật	

2.3 Xác định yêu cầu phi chức năng

1. Hệ thống phải đáp ứng với hàng nghìn yêu cầu đặt vé mỗi ngày
2. Thông tin phim nên được hiển thị rõ ràng, tóm tắt nội dung, hiển thị poster cũng như dàn diễn viên, đạo diễn
3. Hệ thống cần được thiết kế để nâng cấp, mở rộng dễ dàng
4. Cung cấp phương thức nhập dữ liệu nhanh chóng, tiện lợi cho bên quản lý
5. Giao diện người dùng bắt mắt, nhưng cũng đồng thời dễ hiểu, dễ làm quen
6. Có khả năng lưu trữ lượng dữ liệu lớn lâu dài
7. Hệ thống ổn định, mượt mà

Phần 3. Phân tích

3.1 Phân tích tĩnh

3.1.1 Xác định lớp

Các danh từ thu được từ các kịch bản là:

Ứng dụng, người dùng, phim, khuyến mãi, hệ thống, vé, đồ ăn, hóa đơn, tài khoả, rạp, lịch chiếu, sơ đồ phòng chiếu, chỗ ngồi, email, nhân viên, ứng dụng, quy trình, QR code, app quản trị, quản trị viên, danh mục, ngày khởi chiếu, giới thiệu phim, diễn viên trong phim, thẻ loại, sơ đồ phòng chiếu

Loại bỏ các danh từ nằm ngoài phạm vi mục đích của hệ thống và các danh từ hoặc cụm từ trùng lặp và các danh từ làm thuộc tính của lớp như:

Ứng dụng, QR code, email, hệ thống, quy trình, app quản trị, danh mục, ngày khởi chiếu, giới thiệu phim

Vậy các danh từ sau có thể là ứng viên các lớp thực thể:

Người dùng, phim, khuyến mãi, vé, đồ ăn, hóa đơn, tài khoả, rạp, lịch chiếu, sơ đồ phòng chiếu, chỗ ngồi, nhân viên, quản trị viên, diễn viên trong phim, thẻ loại, sơ đồ phòng chiếu

Từ đó ta xây dựng biểu đồ lớp.

3.1.2 Quan hệ giữa các lớp

Trước tiên, ta có quan hệ giữa các lớp thuộc cùng một nhóm nghiệp vụ:

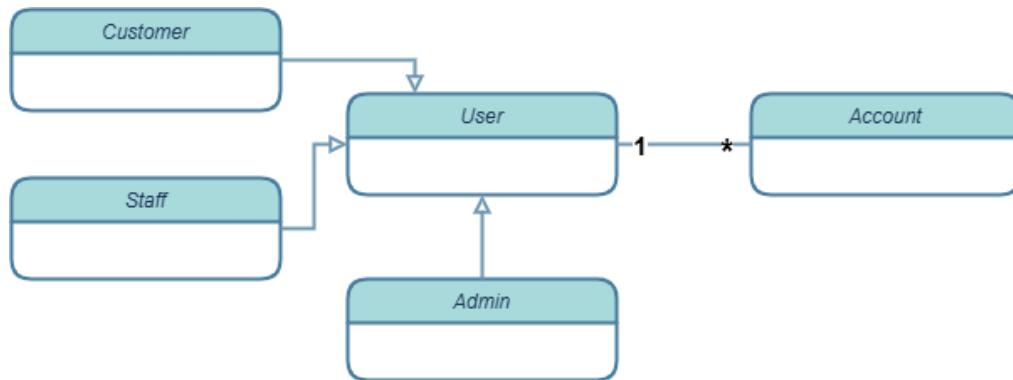


Figure 31 Các lớp người dùng và tài khoản

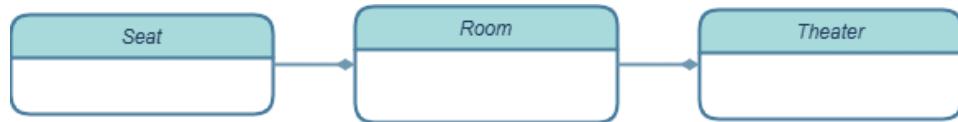


Figure 32 Các lớp cơ sở vật lý rạp chiếu

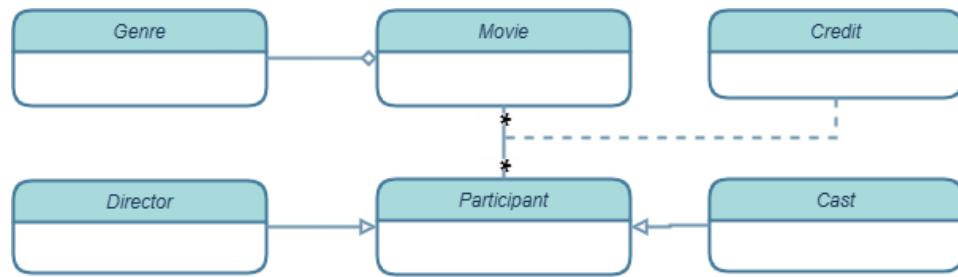


Figure 33 Các lớp liên quan đến phim



Figure 34 Các lớp sản phẩm bán trong rạp chiếu

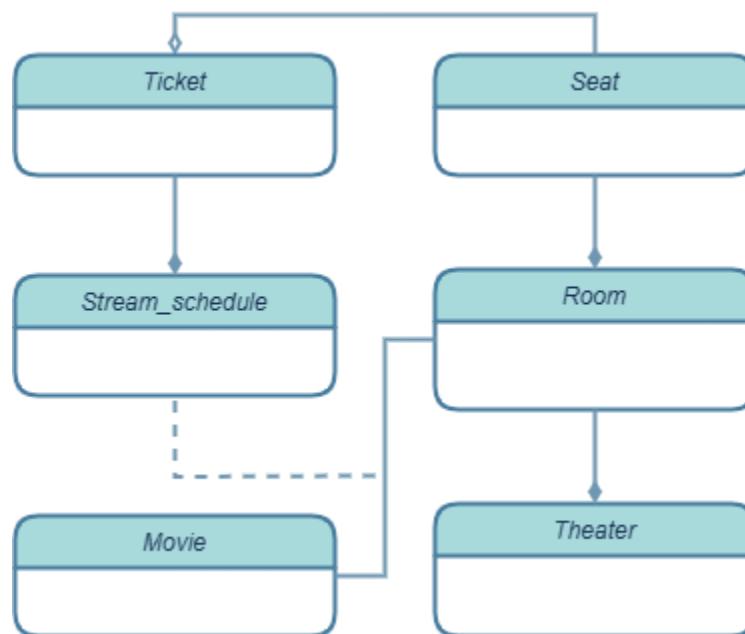


Figure 35 Lớp lịch chiếu, vé xem phim

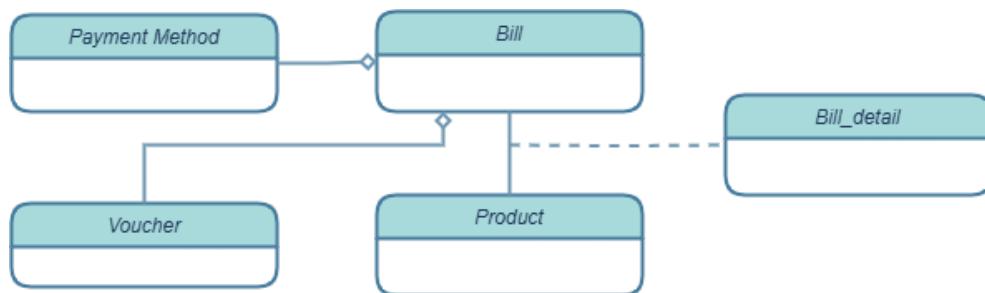


Figure 36 Các lớp liên quan đến hóa đơn

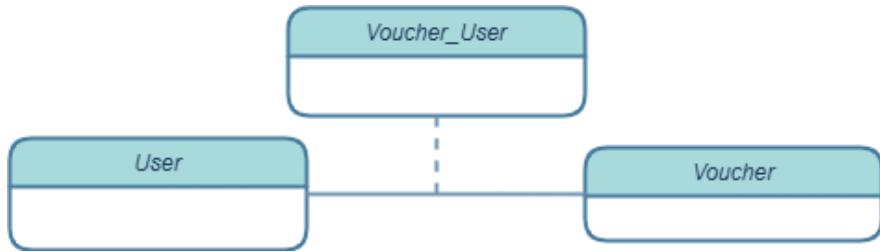


Figure 37 Llop liên kết người dùng và mã giảm giá

Từ đó, ta có sơ đồ quan hệ các lớp thực thể:

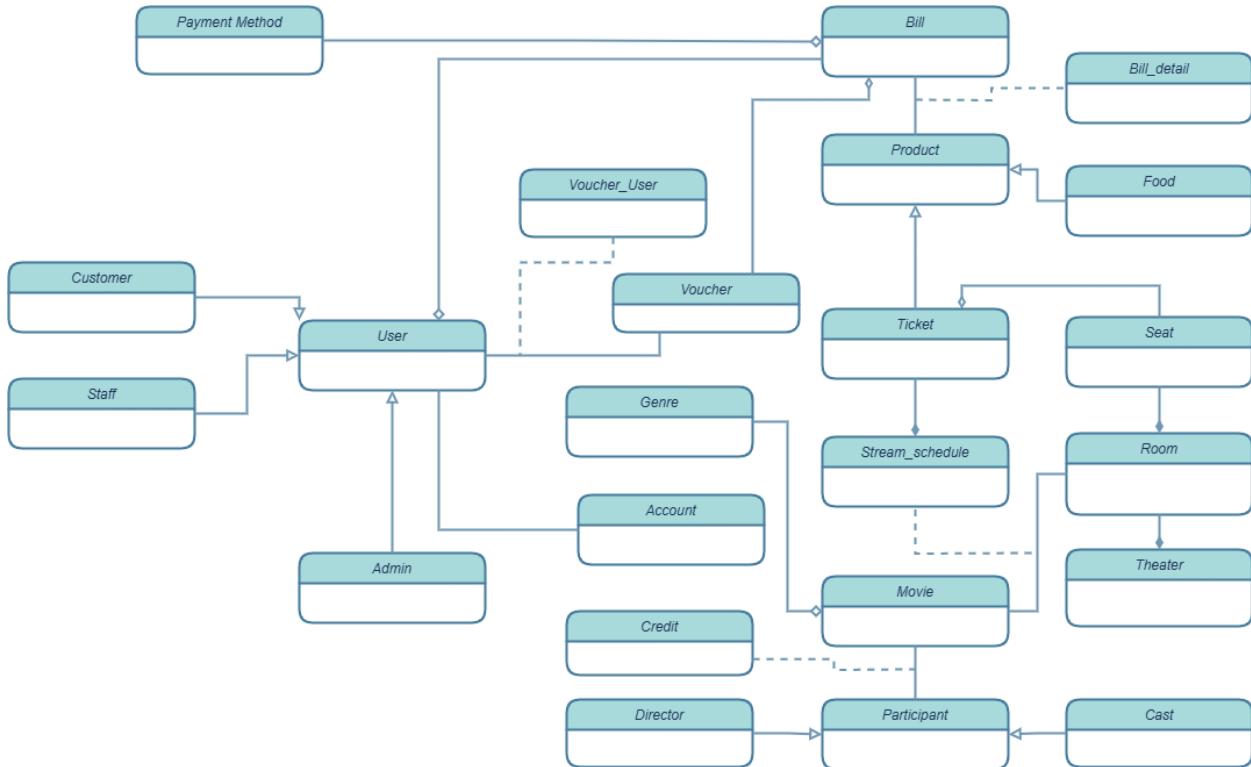


Figure 38 Quan hệ giữa các lớp

3.1.3 Xây dựng biểu đồ lớp

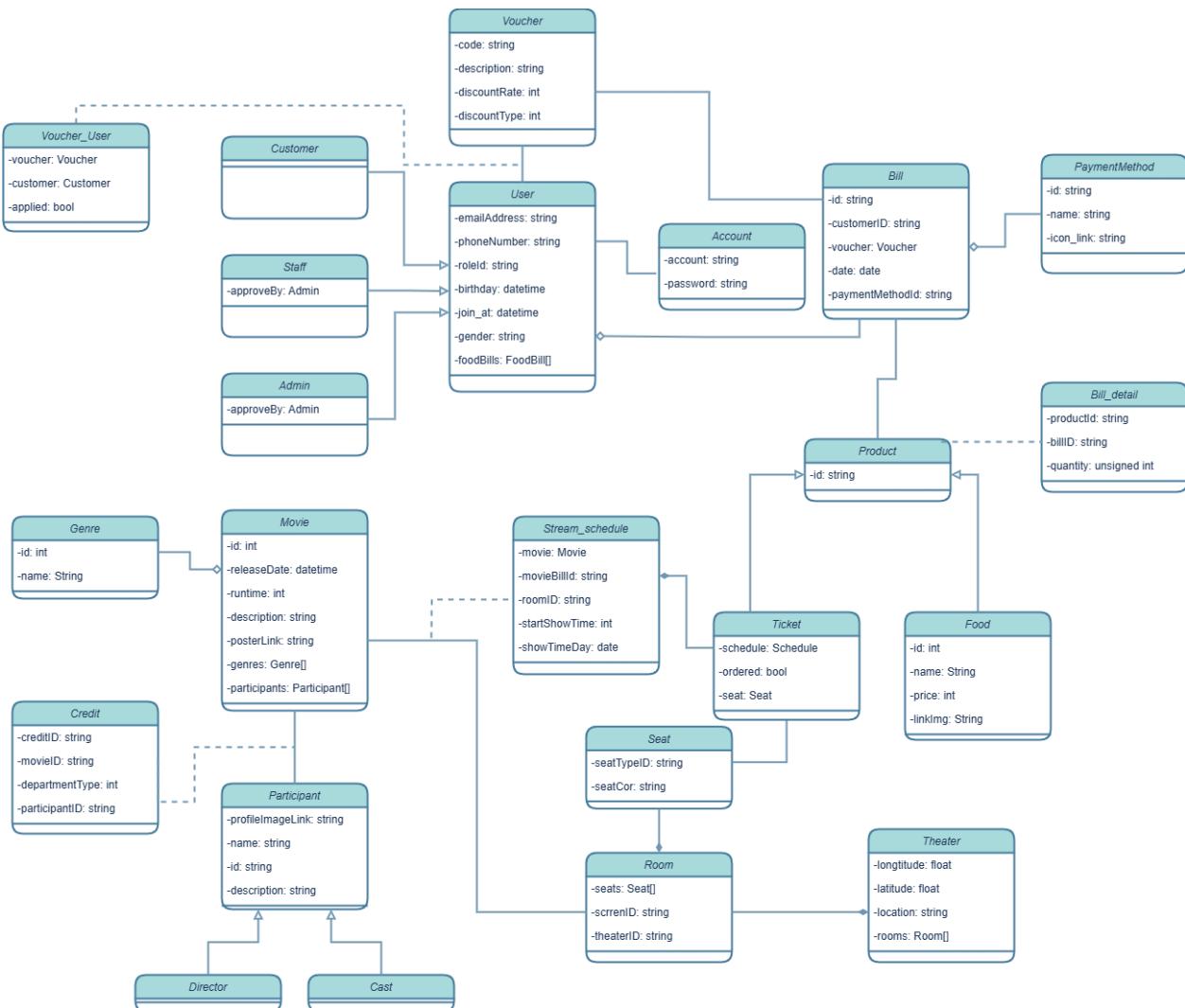


Figure 39 Xây dựng biểu đồ lớp

3.2 Phân tích động

3.2.1 Hiện thực hóa usecase bằng biểu đồ tuần tự

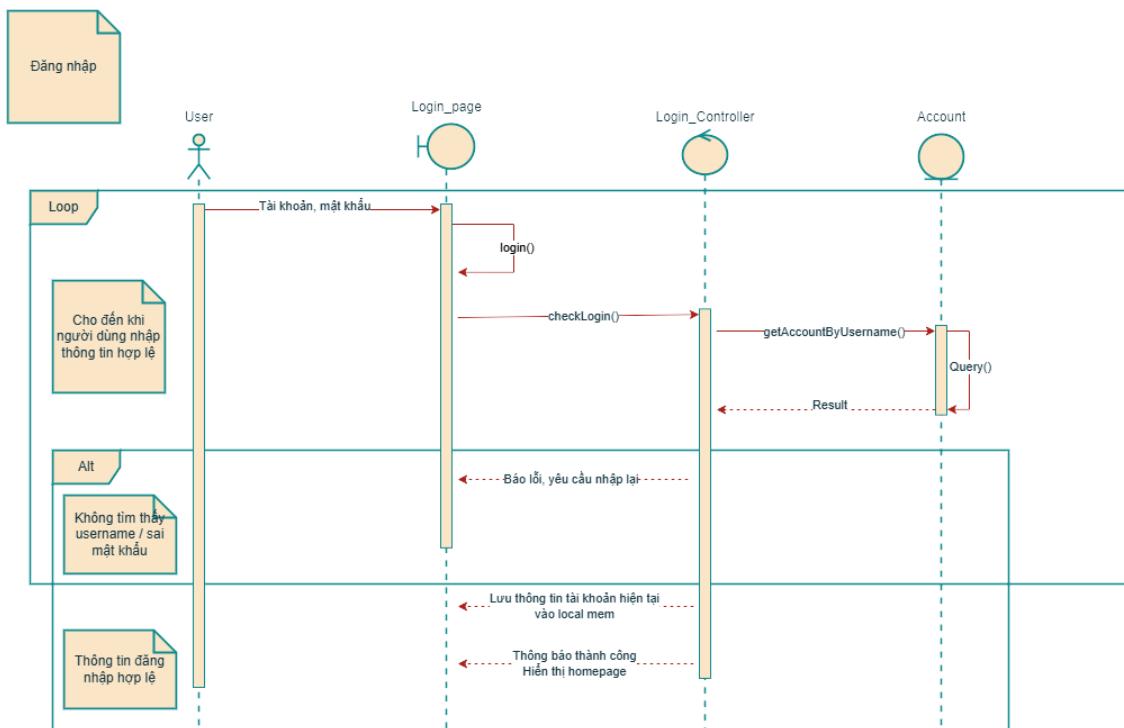


Figure 40 Đăng nhập

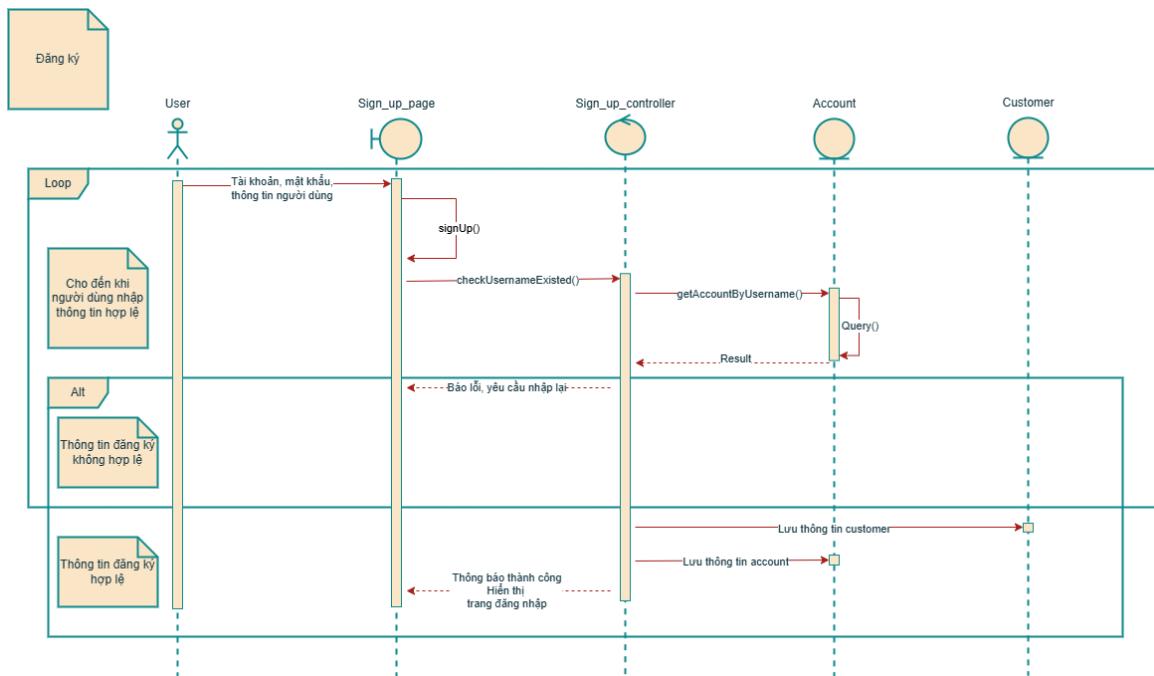


Figure 41 Đăng ký

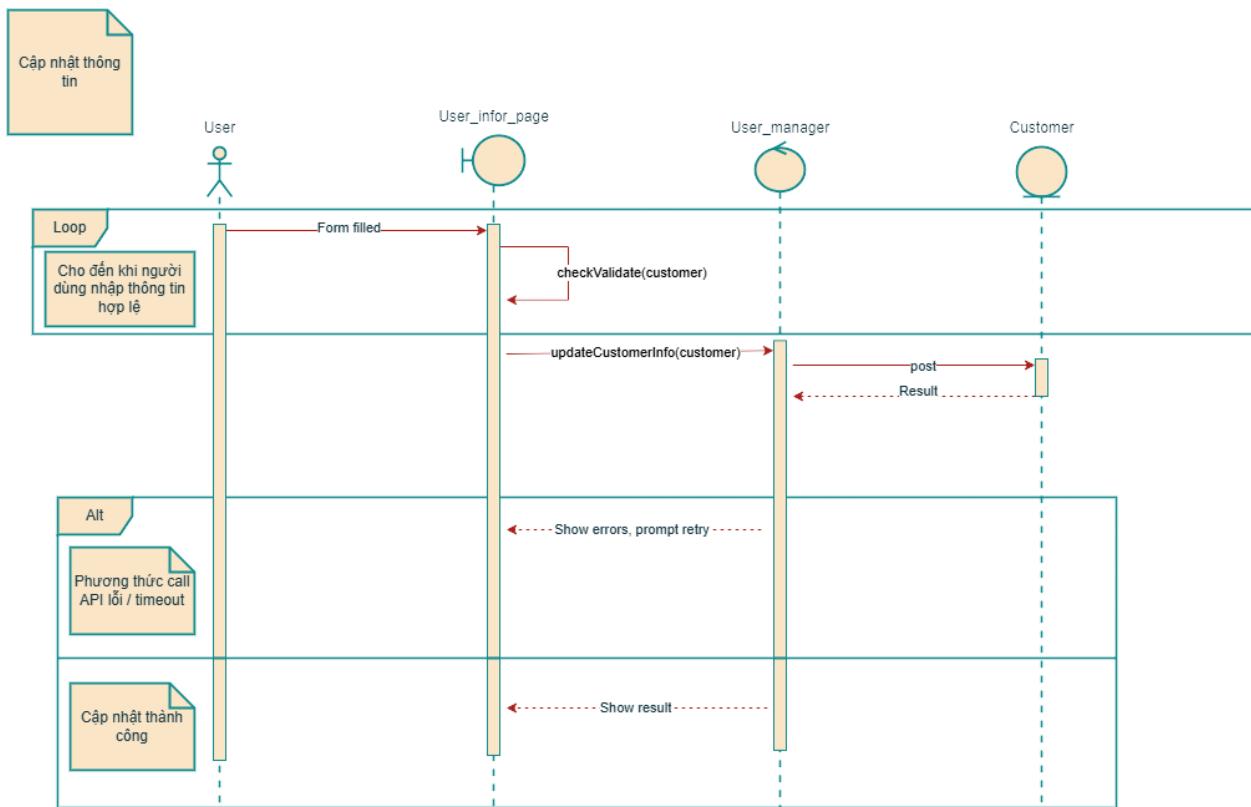


Figure 42 Chỉnh sửa thông tin

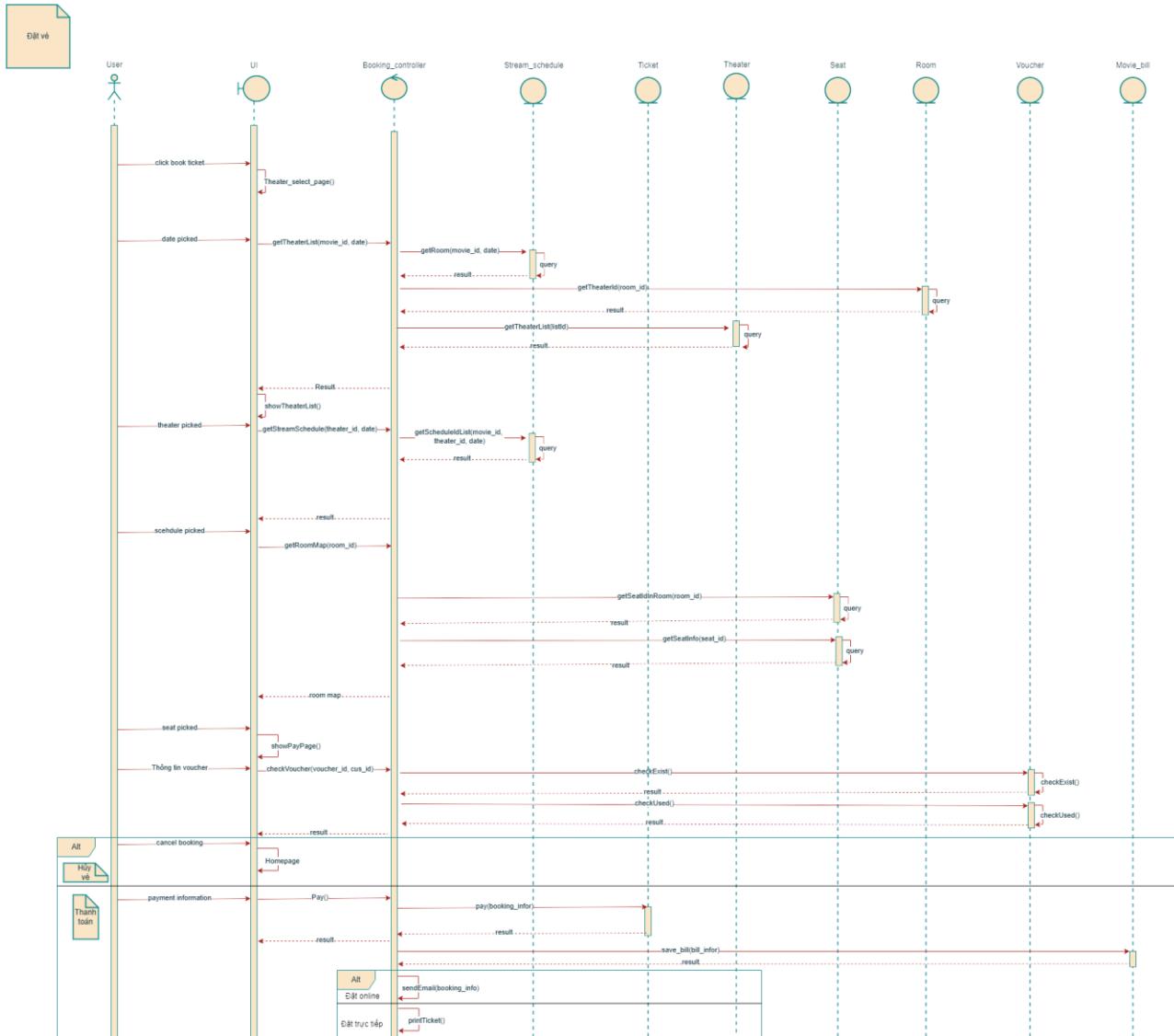


Figure 43 Đặt vé xem phim

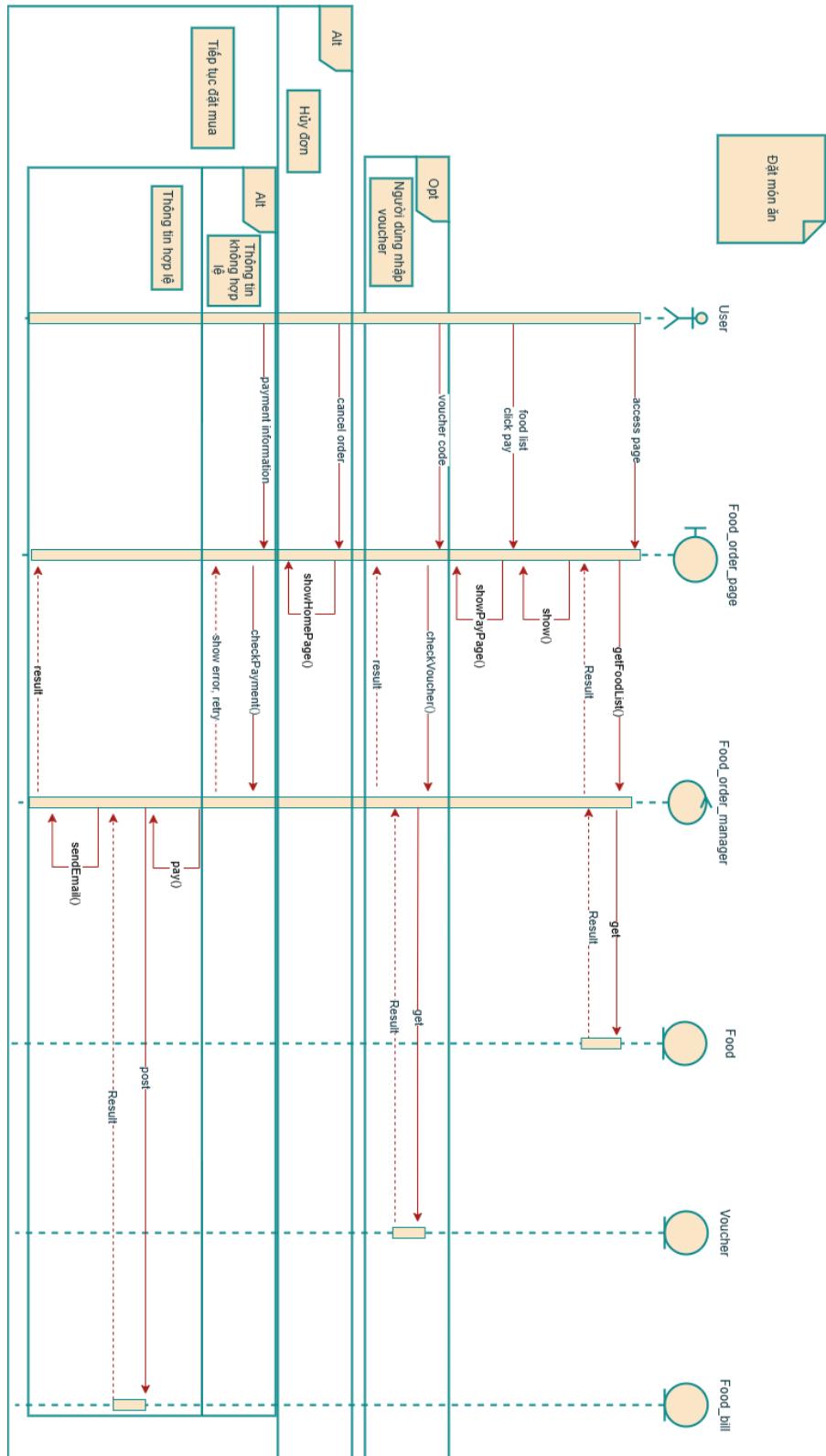


Figure 44 Đặt đồ ăn

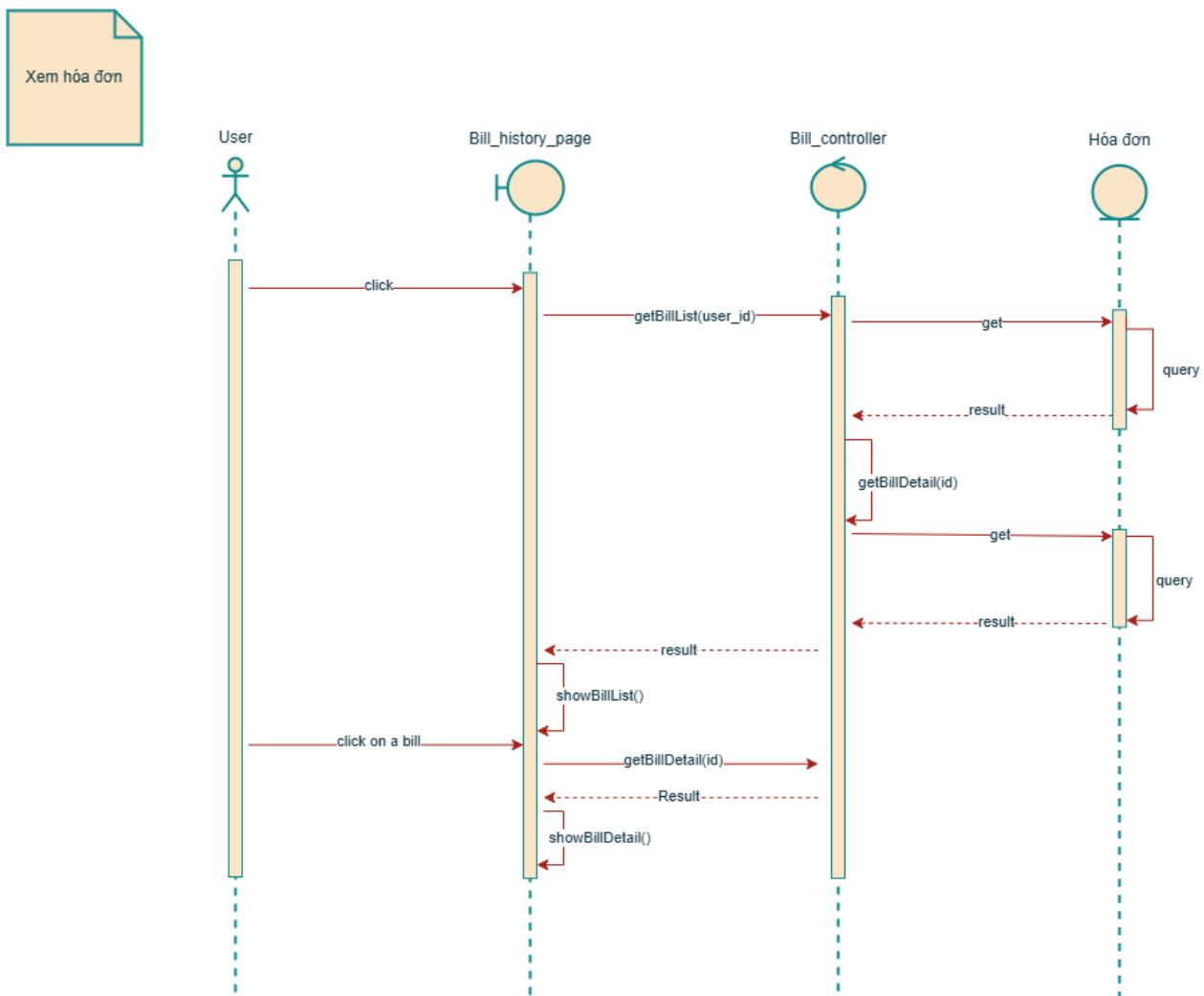


Figure 45 Xem hóa đơn

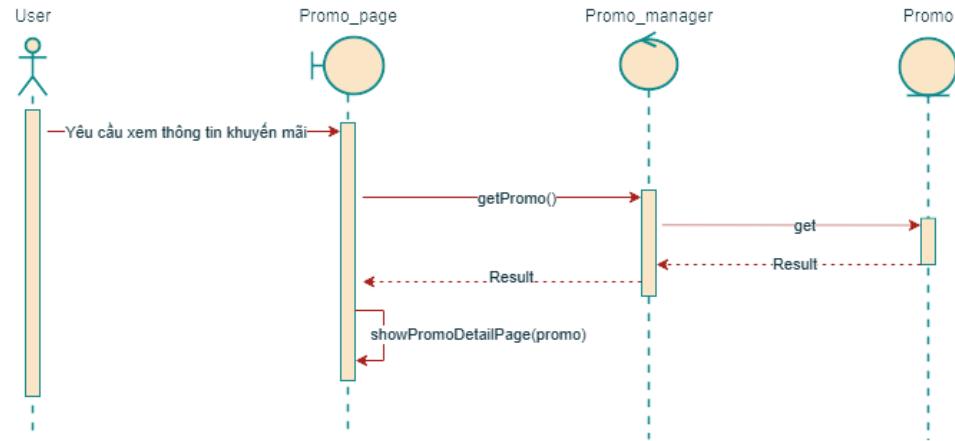


Figure 46 Xem thông tin khuyến mãi

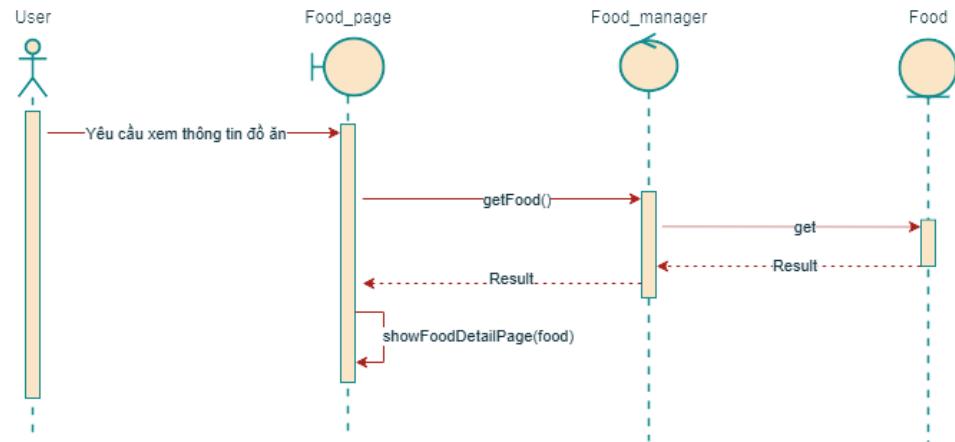


Figure 47 Xem chi tiết đồ ăn

Xem thông tin phim

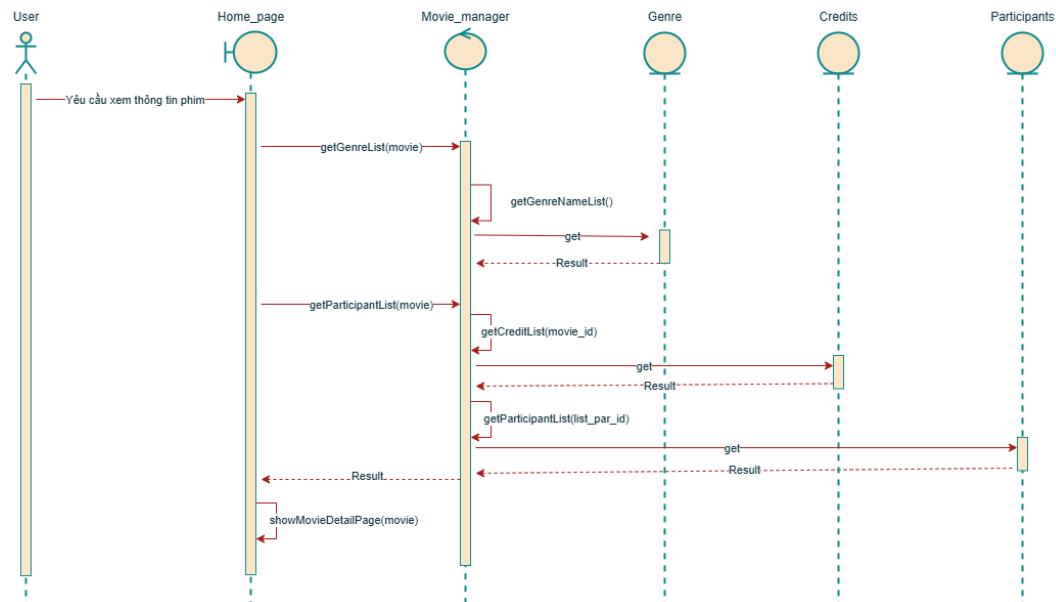


Figure 48 Xem chi tiết phim

Tạo QR thanh toán

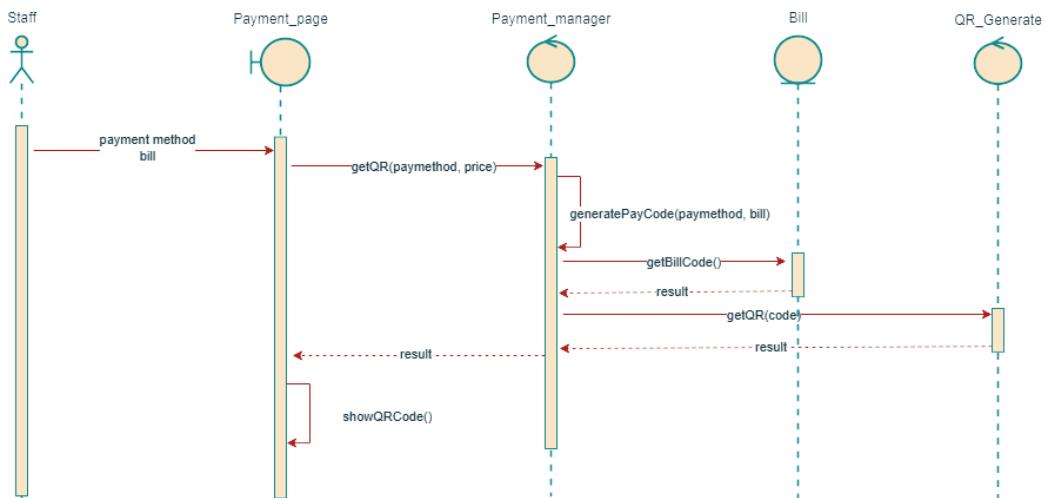


Figure 49 Tạo QR thanh toán

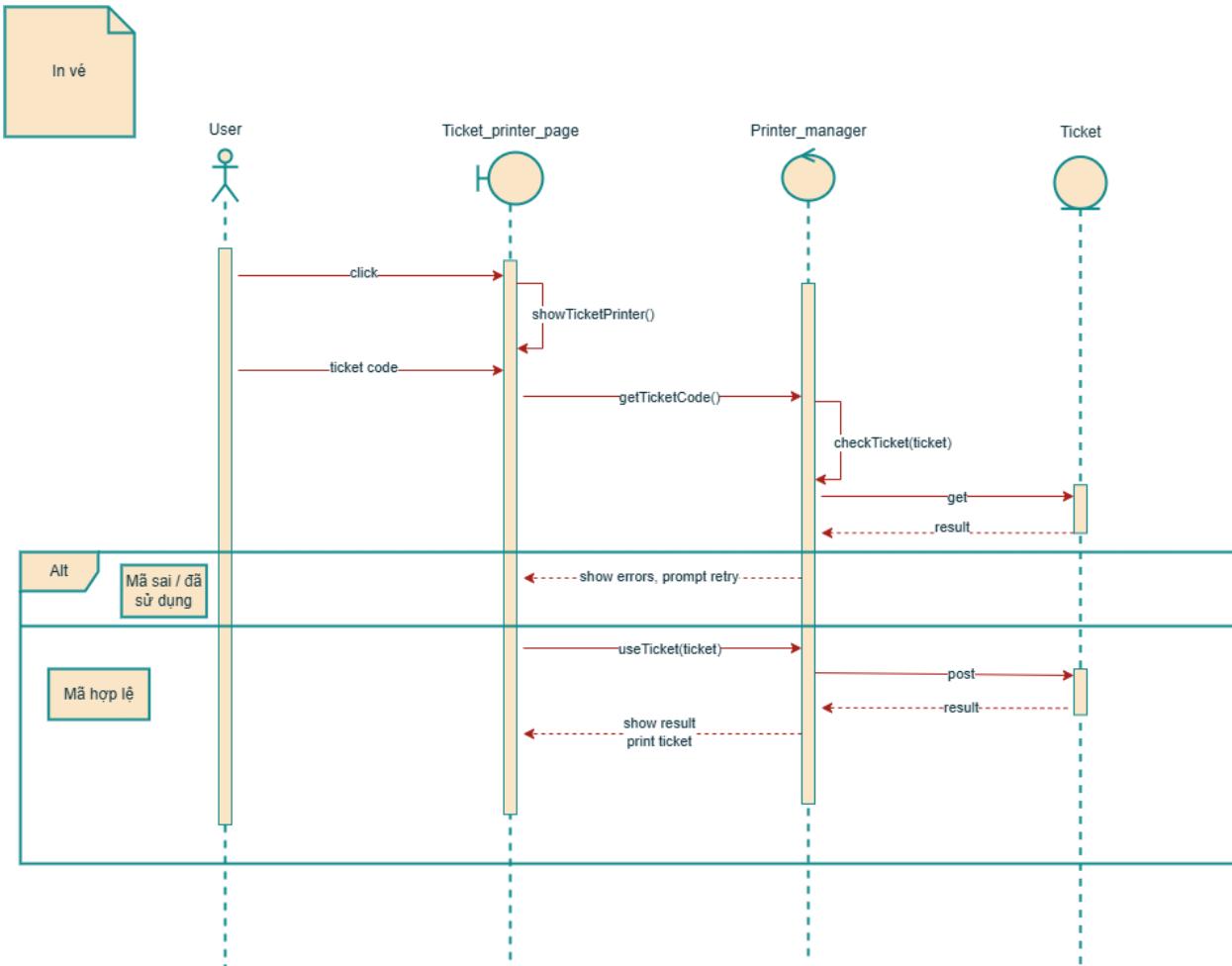


Figure 50 In vé

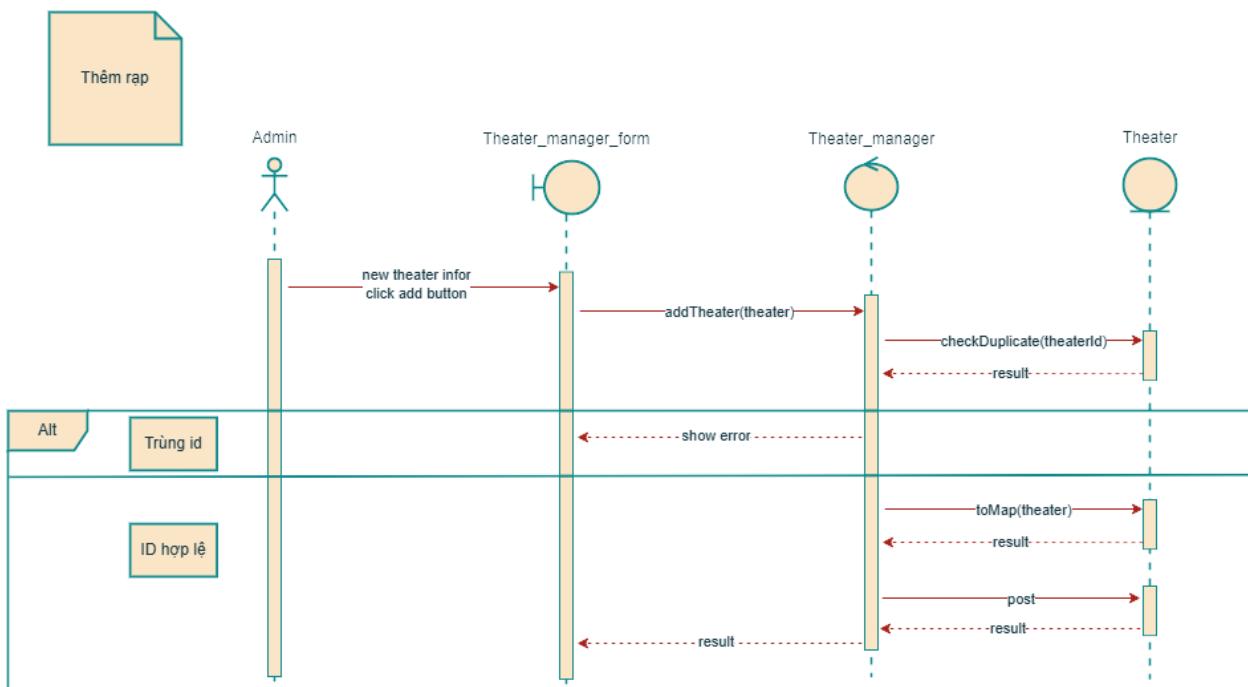


Figure 51 Thêm rạp

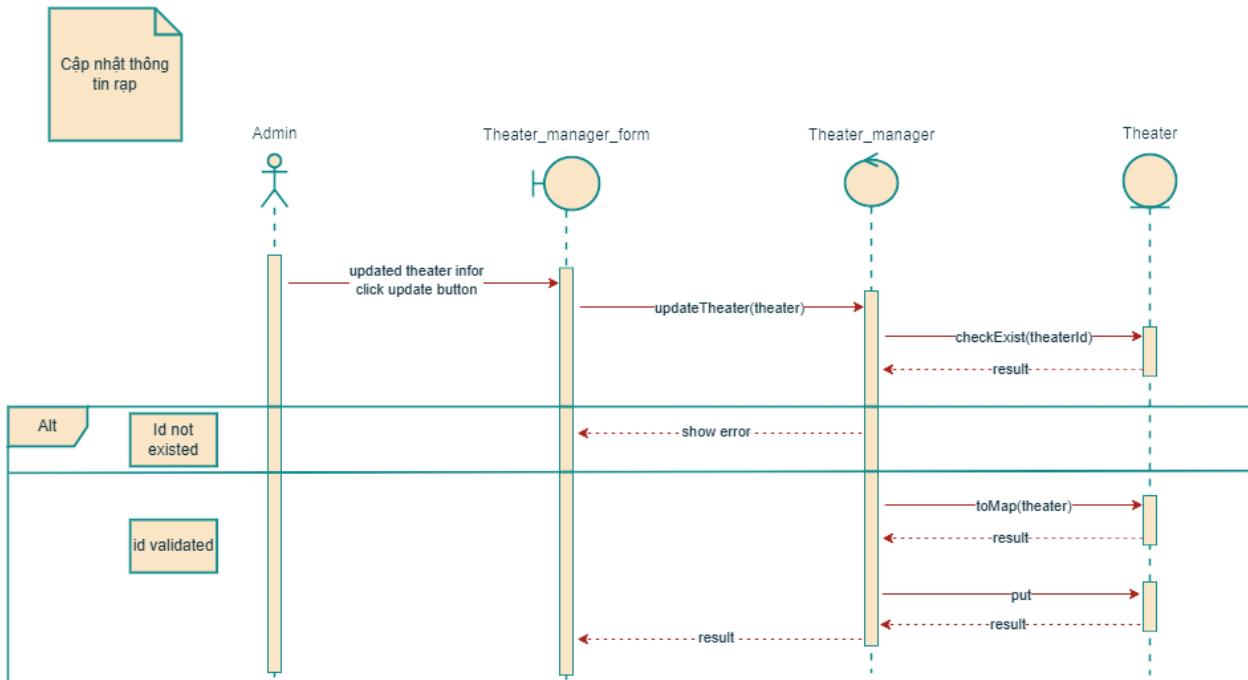


Figure 52 Cập nhật thông tin rạp

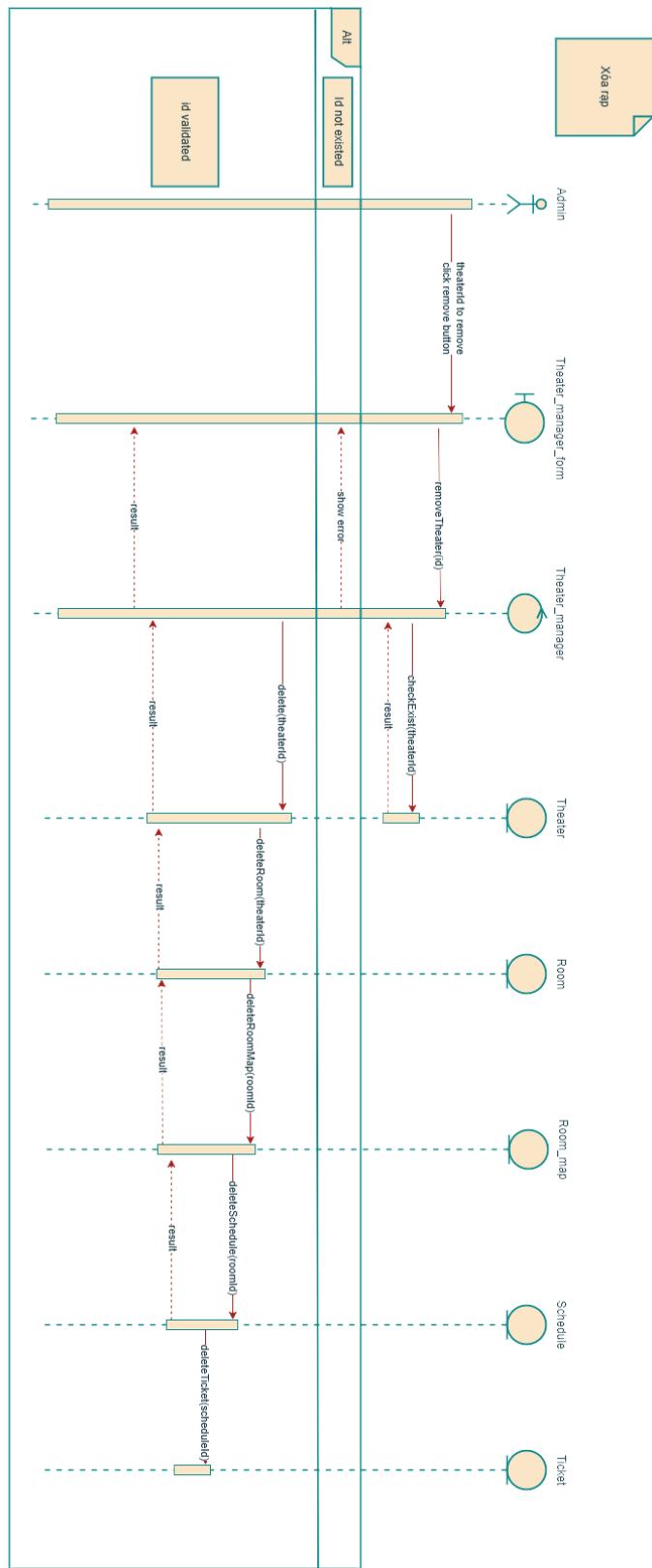


Figure 53 Xóa rap

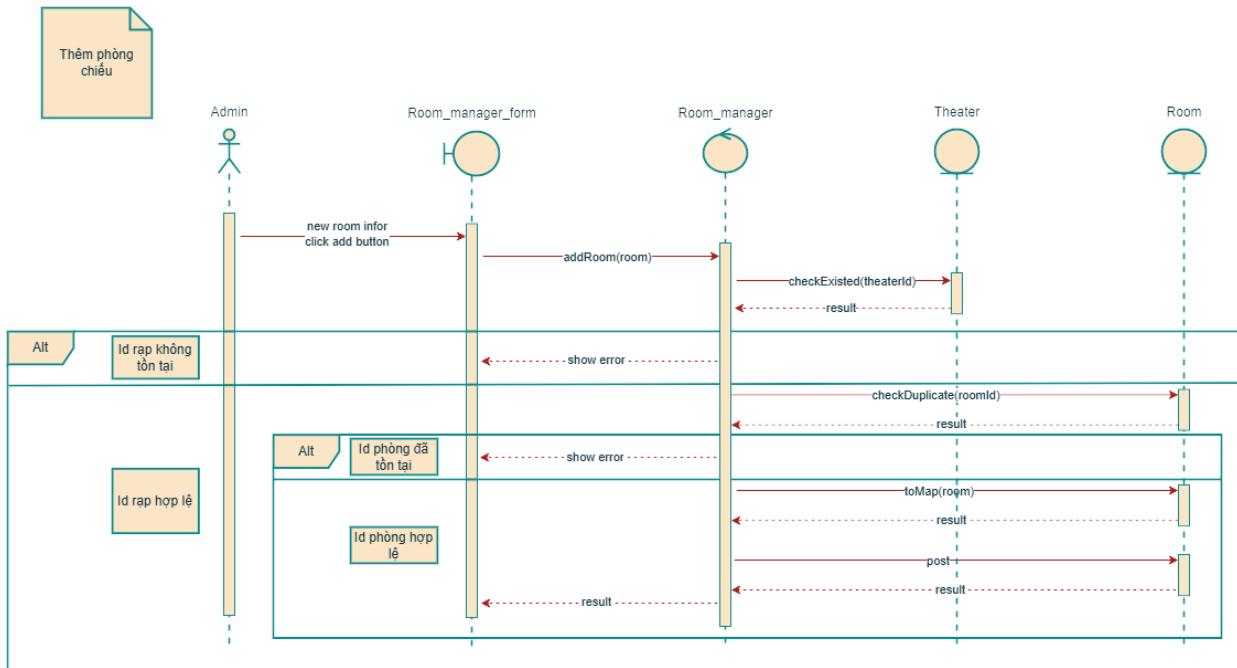


Figure 54 Thêm phòng chiếu

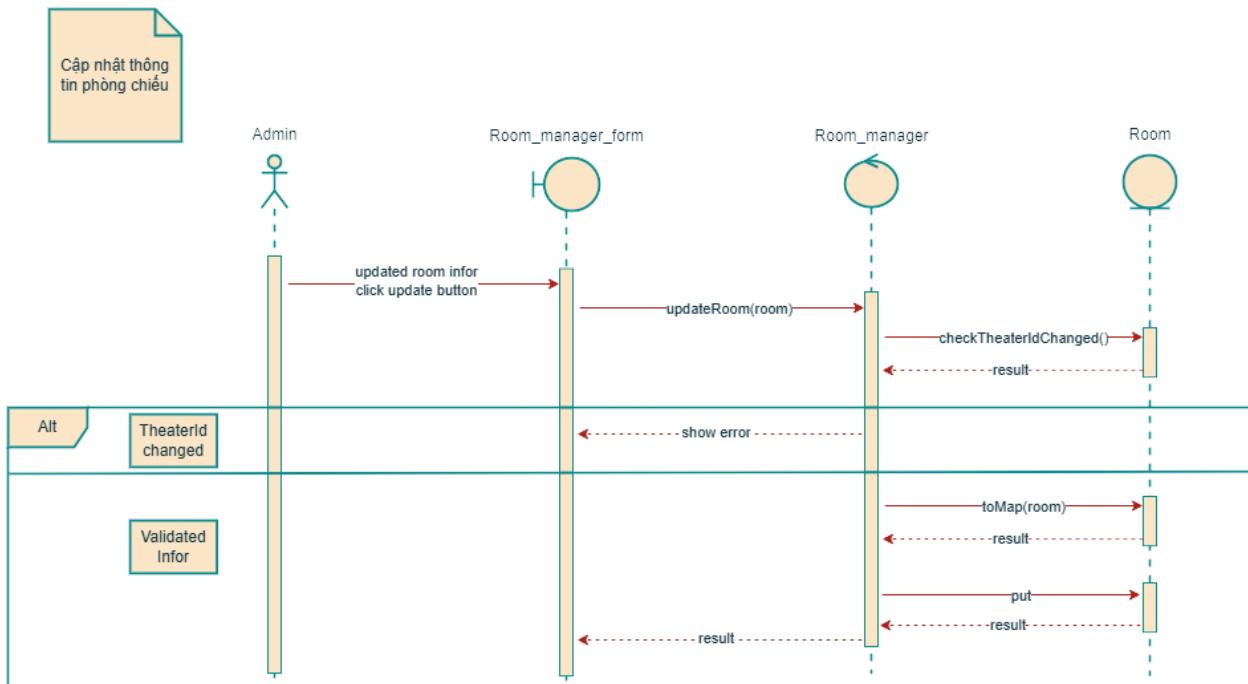


Figure 55 Cập nhật phòng chiếu

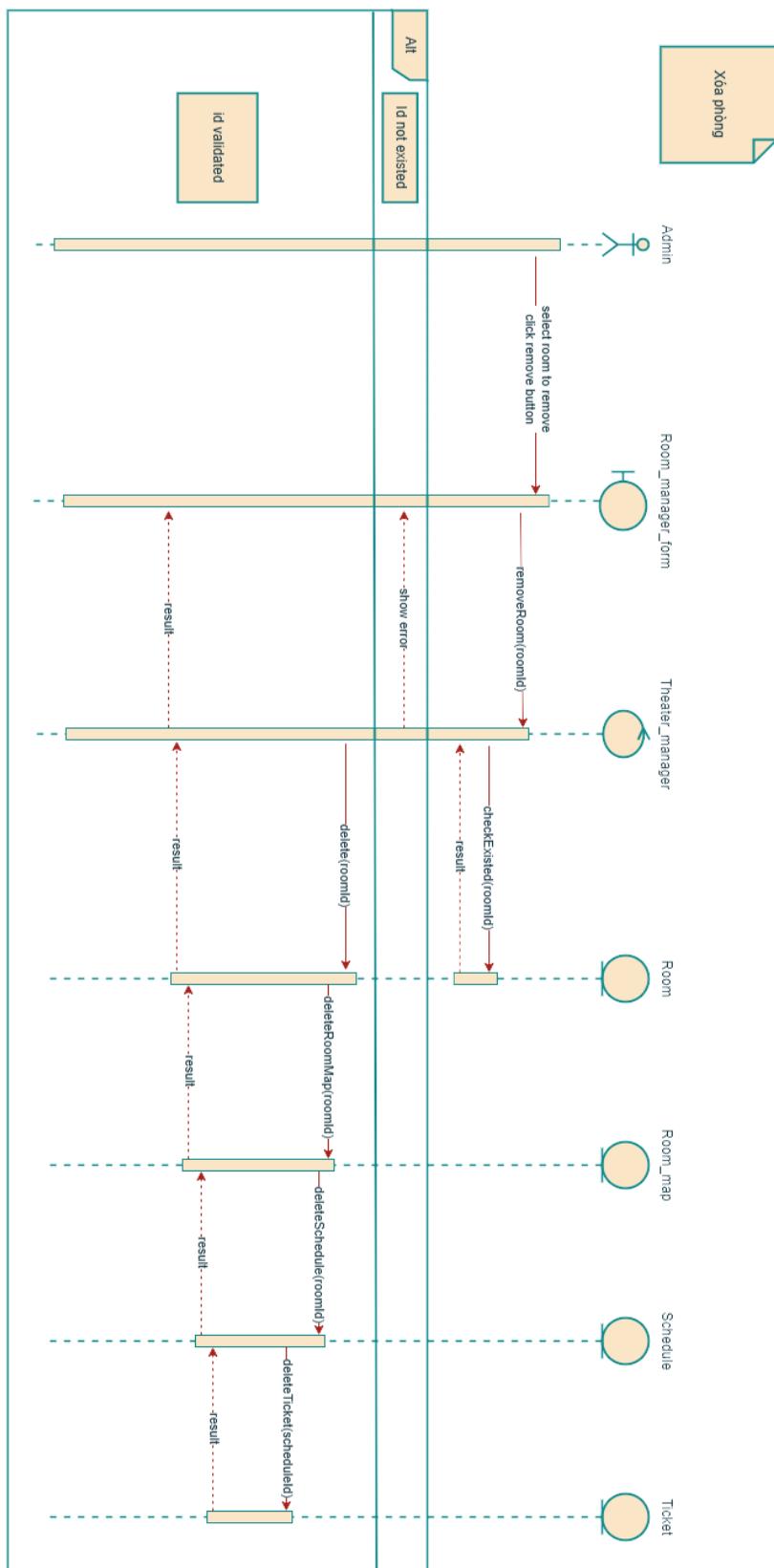


Figure 56 Xóa phòng chiéu

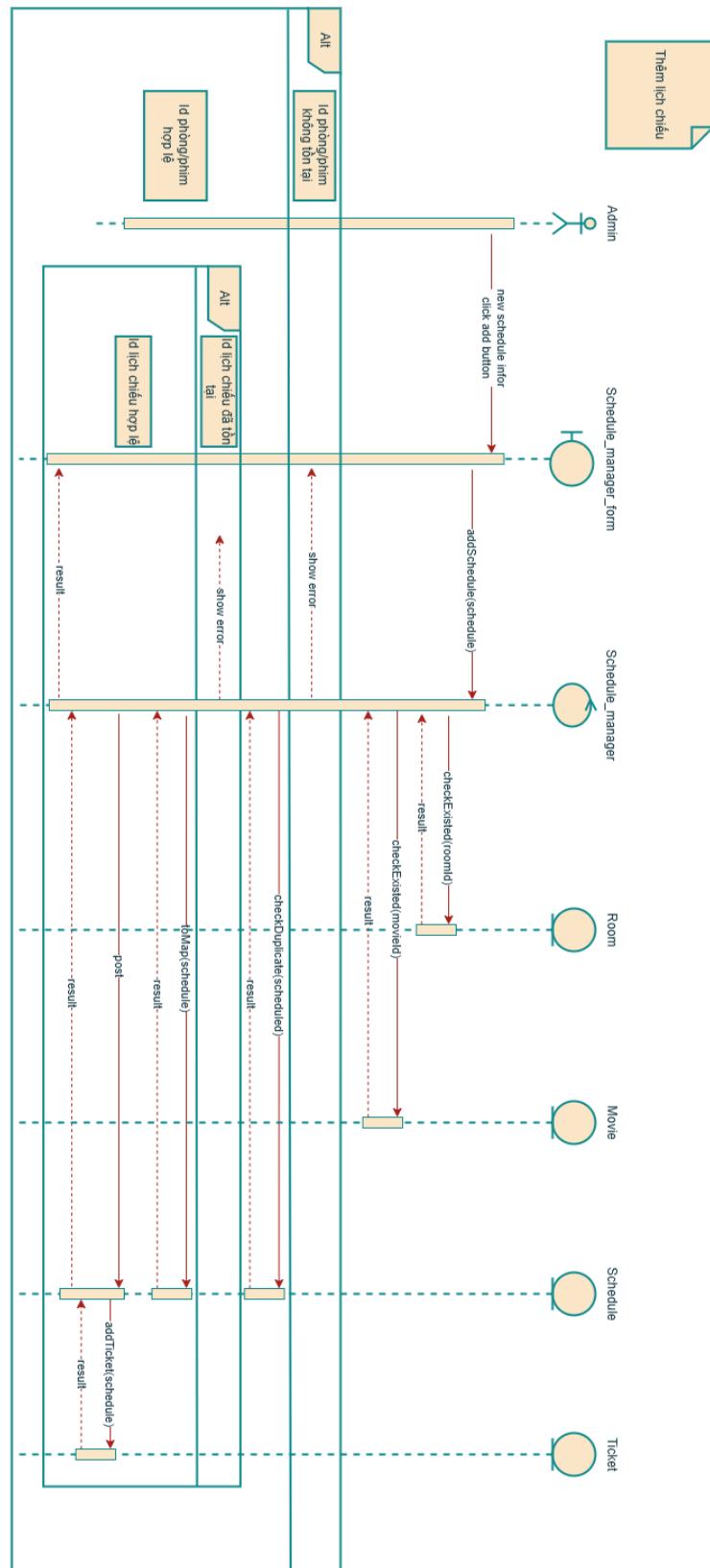


Figure 57 Thêm lịch chiếu

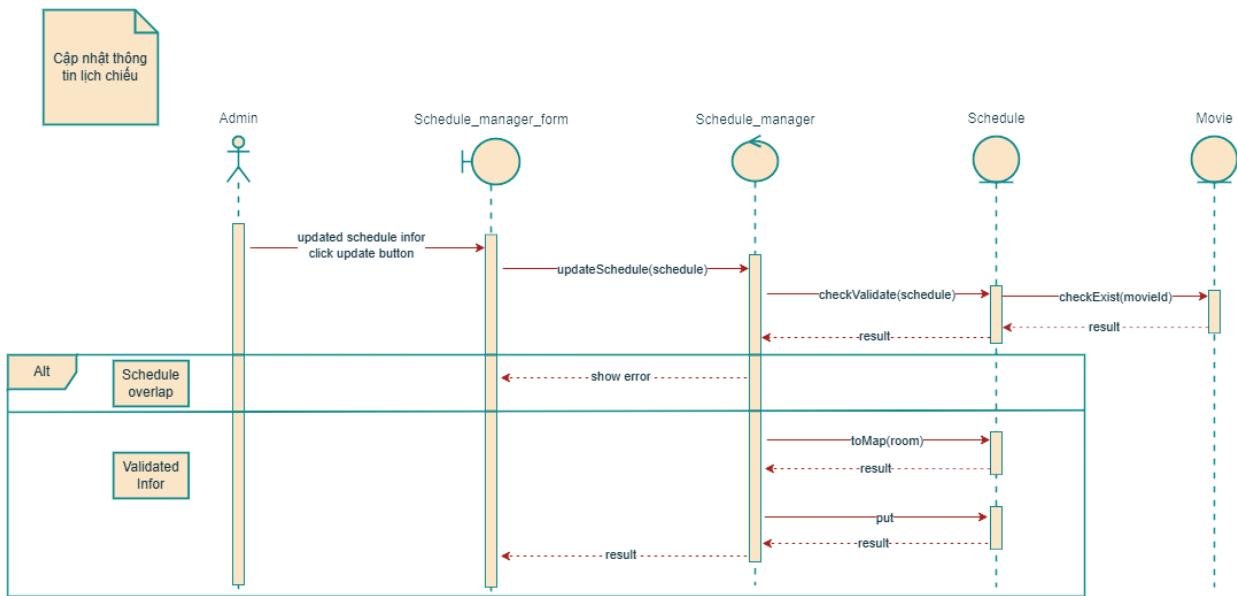


Figure 58 Cập nhật lịch chiếu

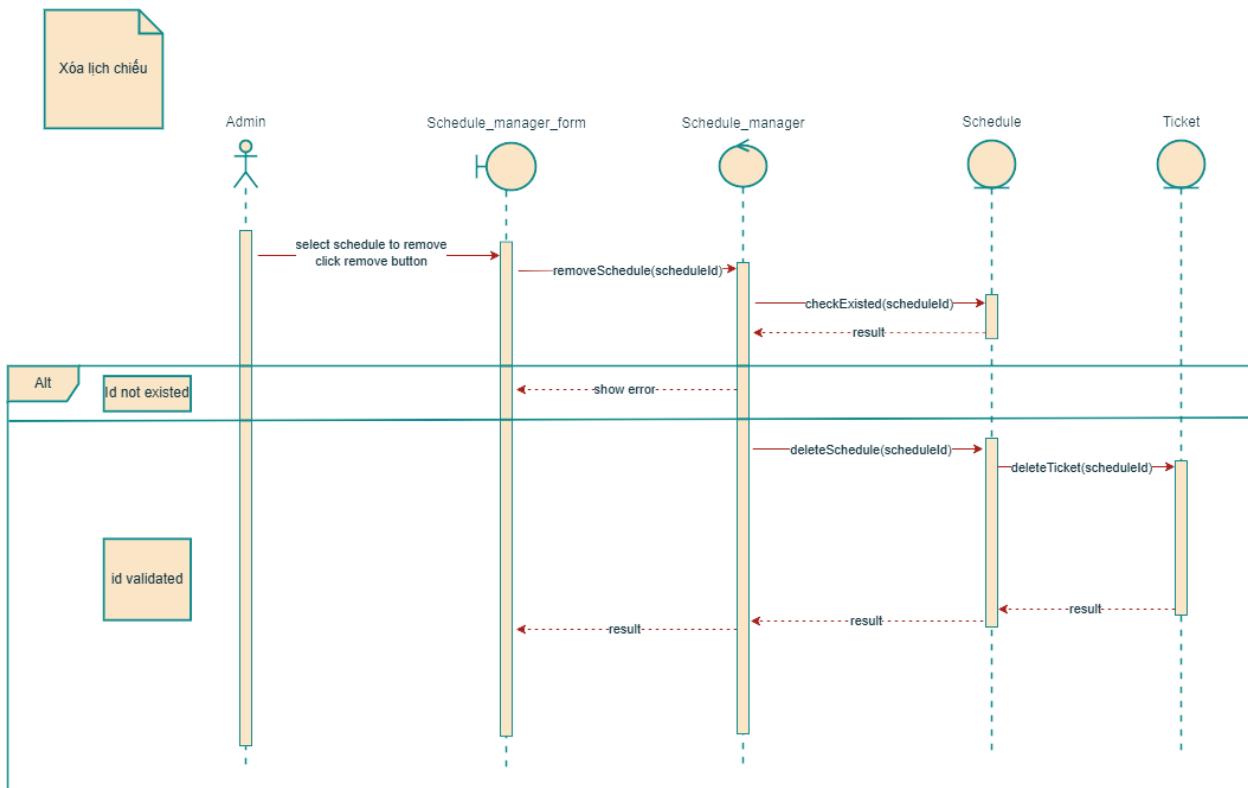


Figure 59 Xóa lịch chiếu

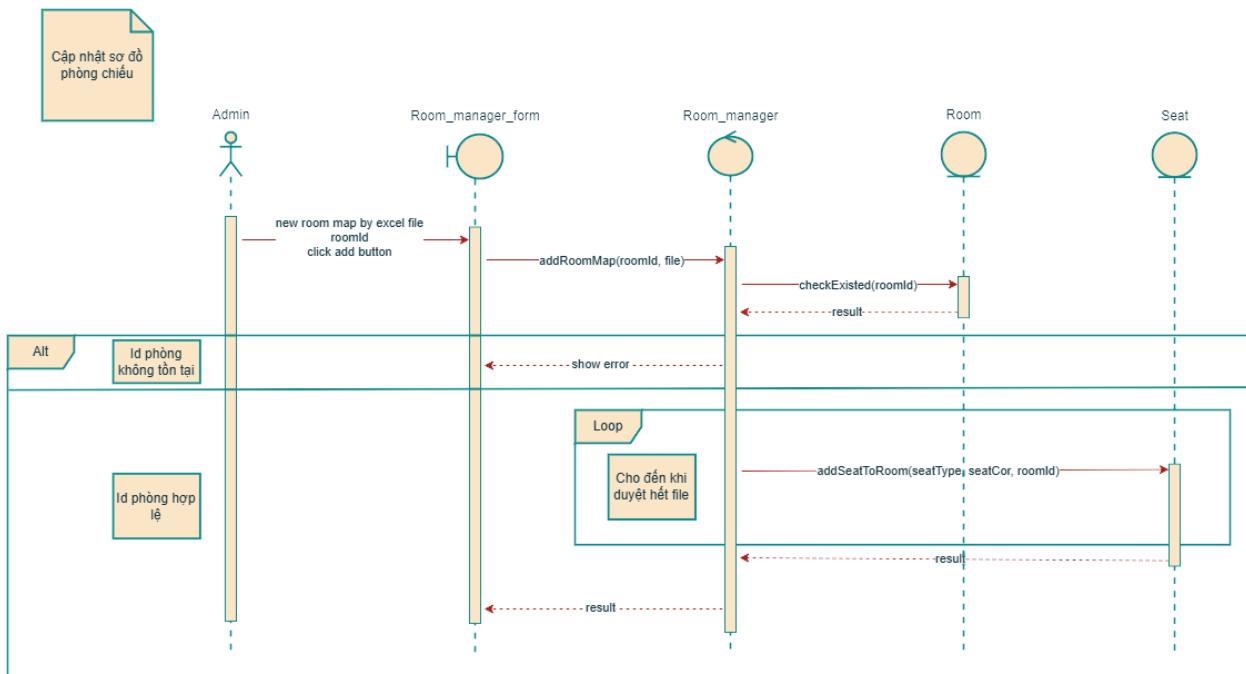


Figure 60 Cập nhật sơ đồ phòng chiếu

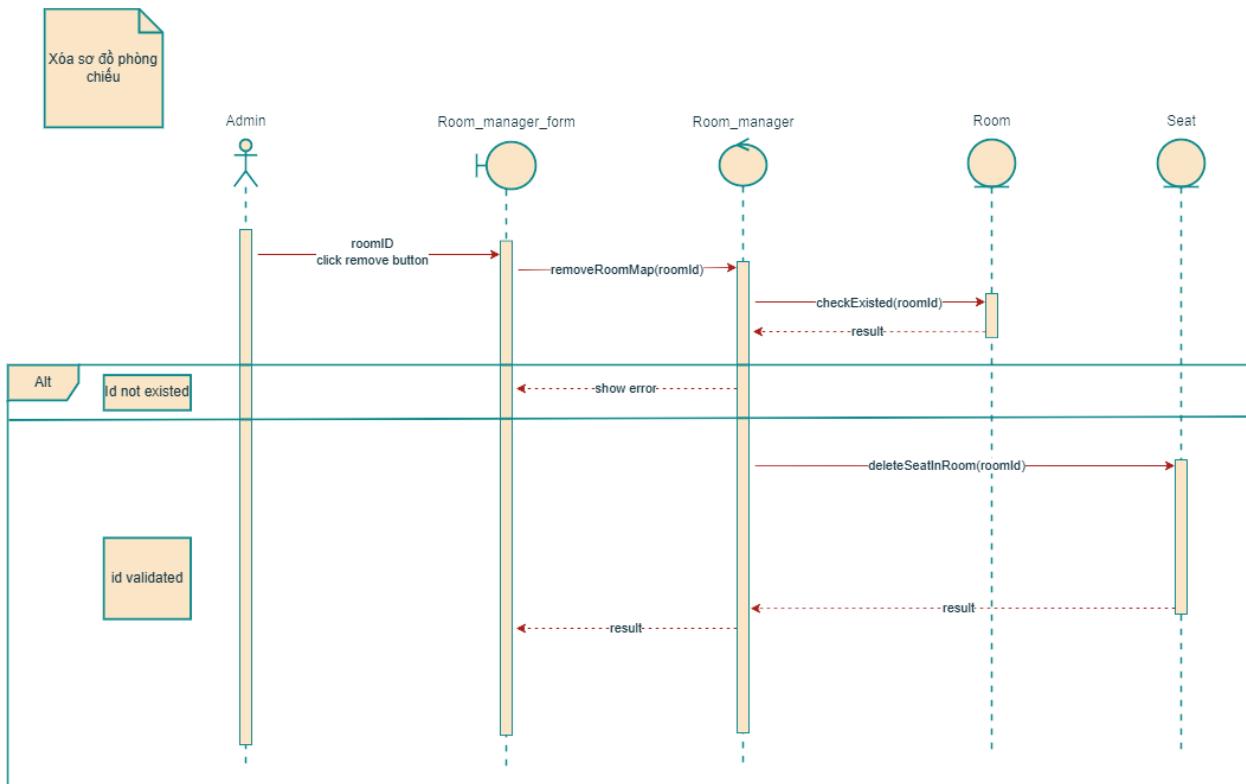


Figure 61 Xóa sơ đồ phòng chiếu

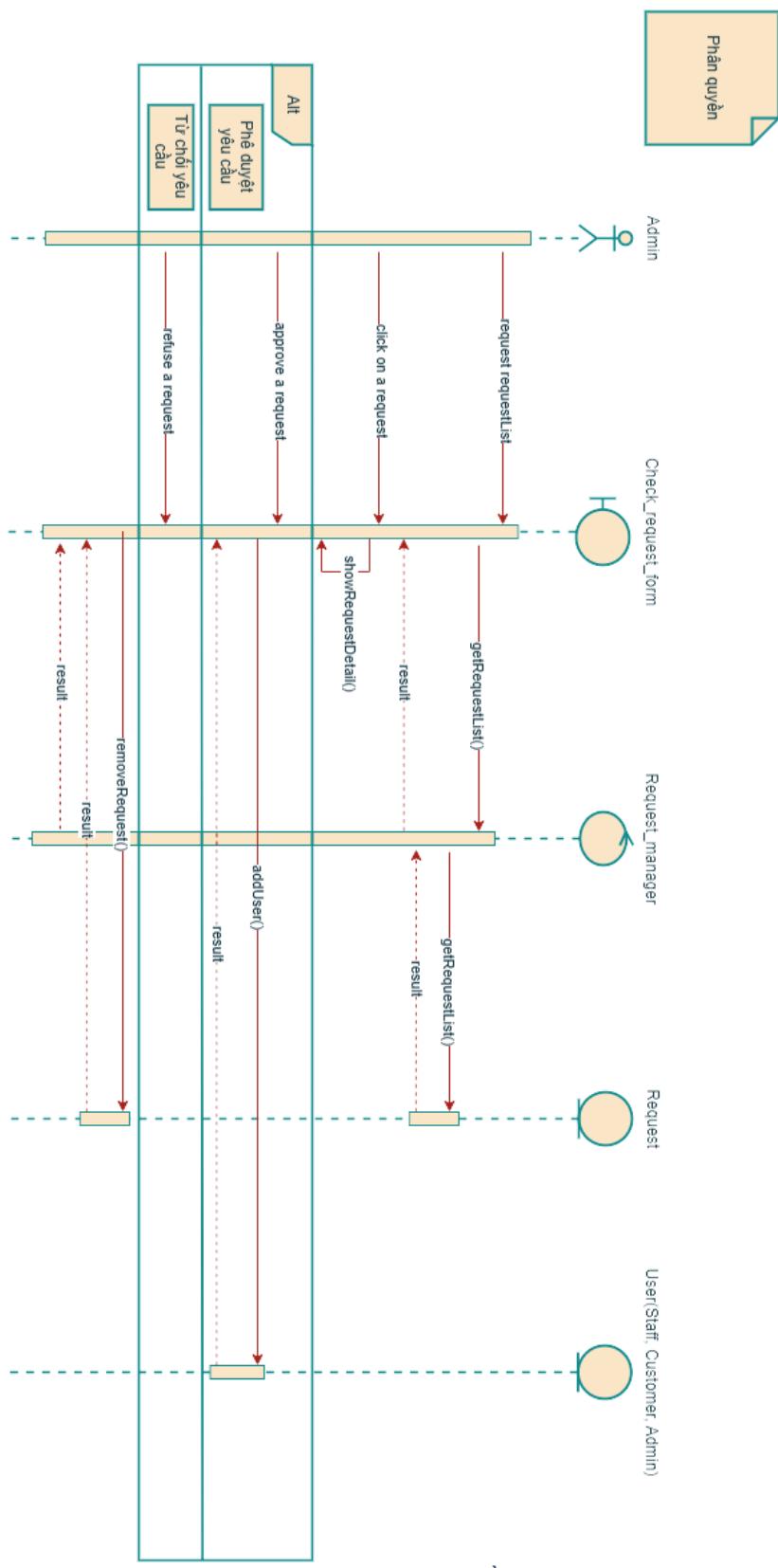


Figure 62 Phân quyền

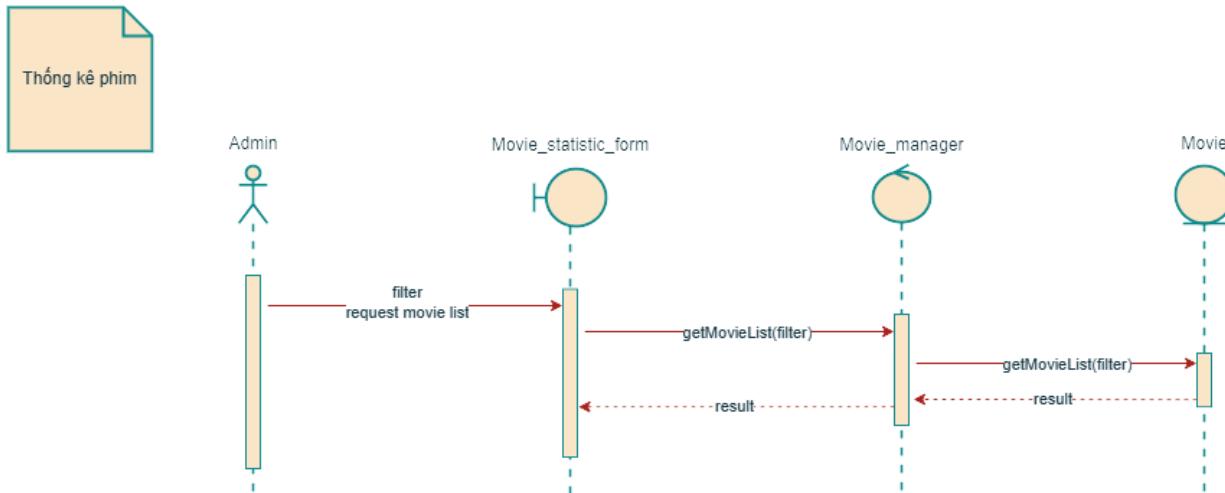


Figure 63 Thống kê phim

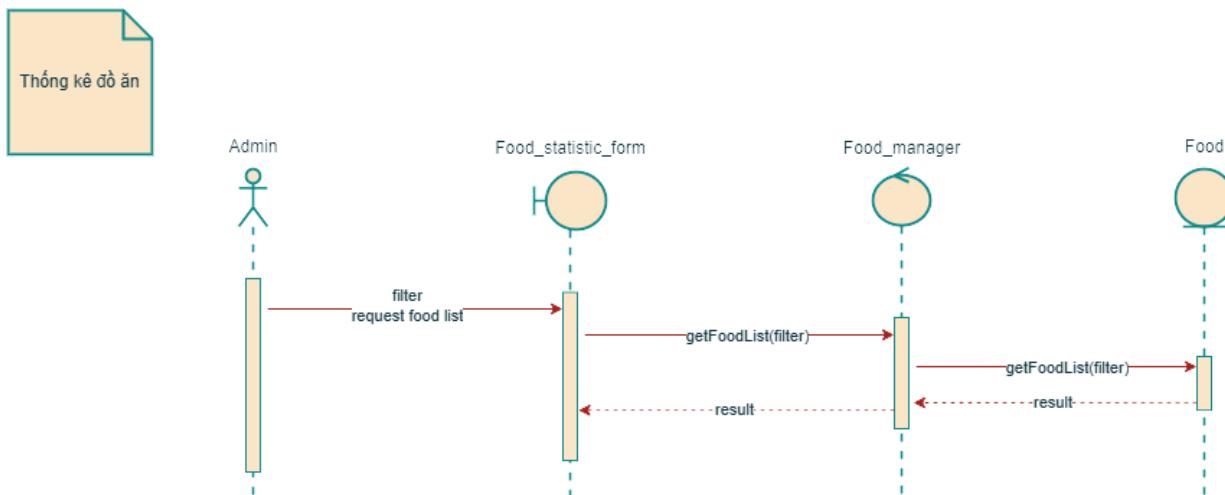


Figure 64 Thống kê đồ ăn

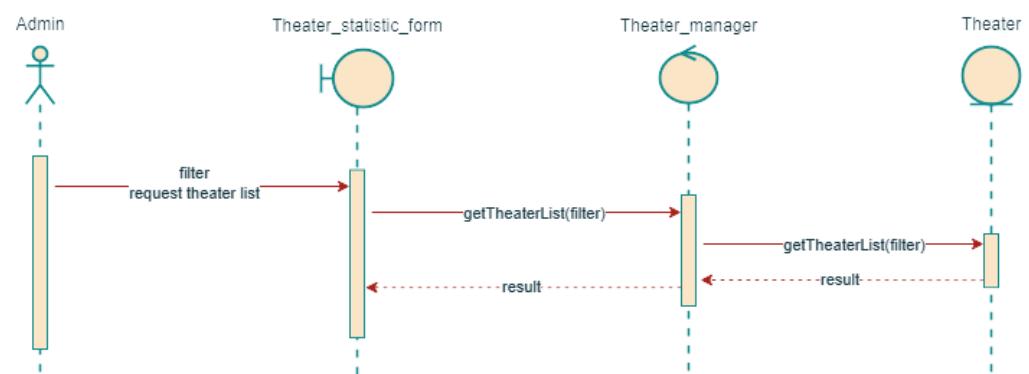


Figure 65 *Thống kê rạp*

3.2.2 Gán phương thức cho các lớp

Từ việc hiện thực hóa usecase, ta gán phương thức cho các lớp:

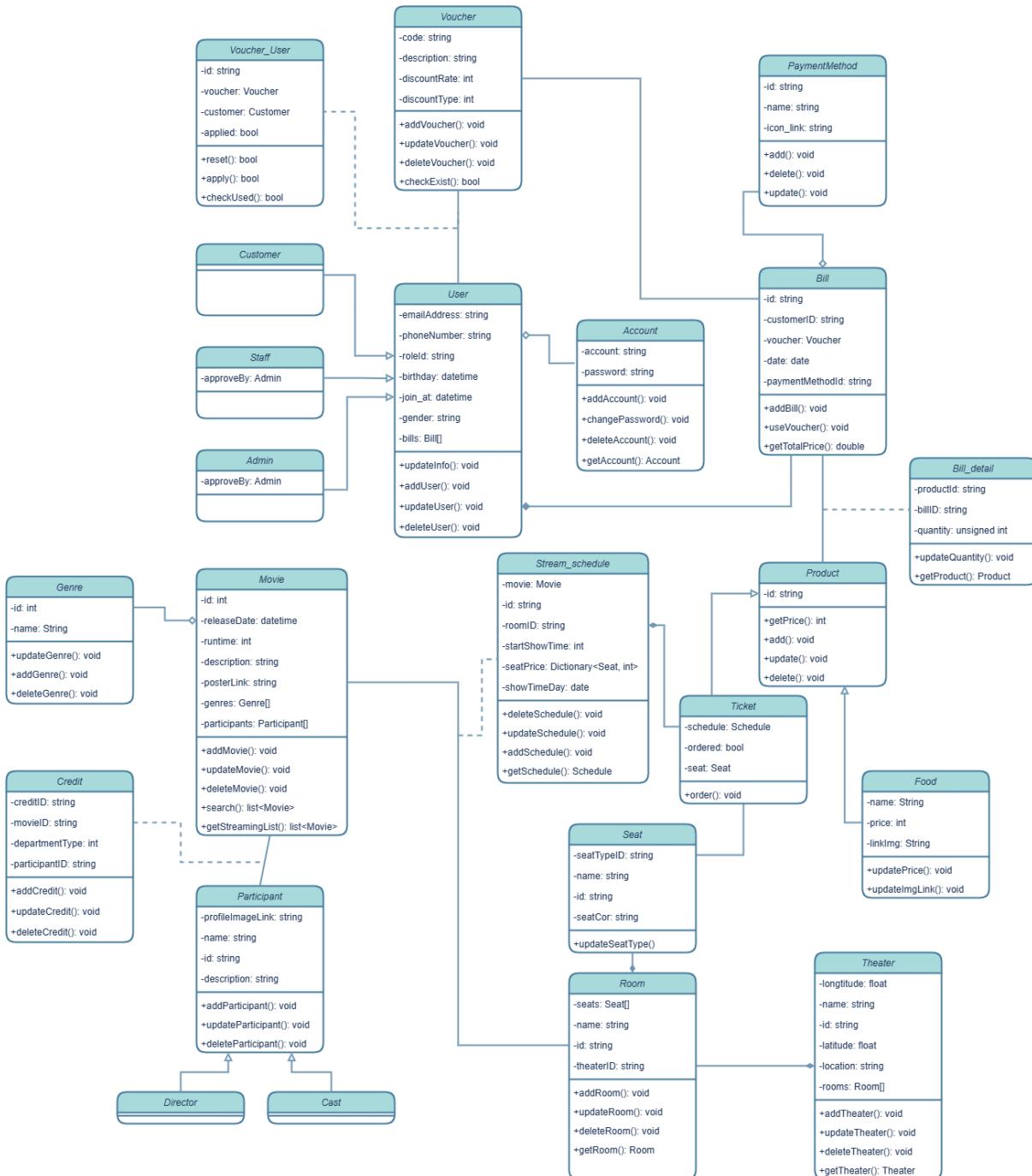
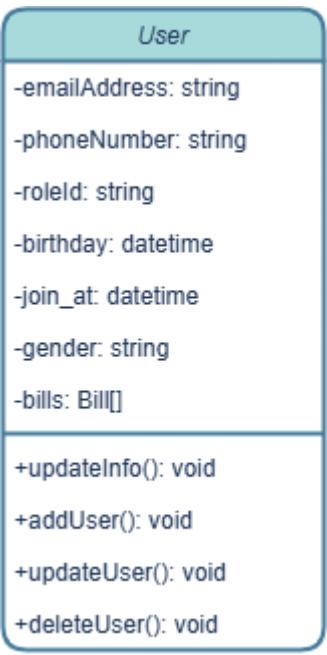
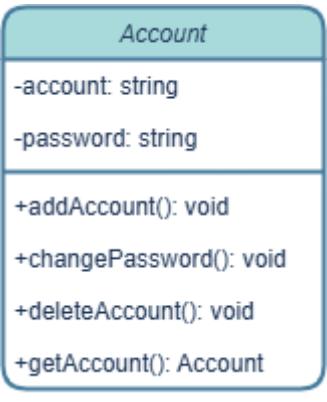
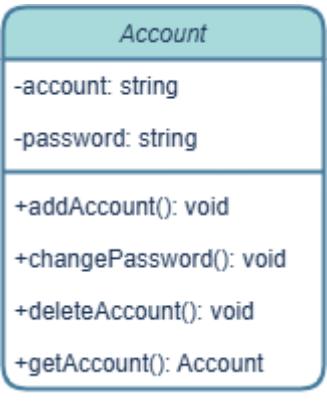


Figure 66 Gán phương thức cho biểu đồ lớp

<pre> Movie -id: int -releaseDate: datetime -runtime: int -description: string -posterLink: string -genres: Genre[] -participants: Participant[] +addMovie(): void +updateMovie(): void +deleteMovie(): void +search(): list<Movie> +getStreamingList(): list<Movie> </pre>	Định nghĩa	Chứa các thông tin cơ bản của đối tượng phim chiếu trong rạp
	Thuộc tính	<ul style="list-style-type: none"> ❖ id: mã phim, mã này là duy nhất ❖ releaseDate: ngày khởi chiếu ❖ runtime: thời lượng phim (phút) ❖ description: mô tả phim ❖ posterLink: đường dẫn đến poster phim ❖ genres: các thể loại phim ❖ participants: người tham gia, gồm đạo diễn và diễn viên
<pre> Genre -id: int -name: String +updateGenre(): void +addGenre(): void +deleteGenre(): void </pre>	Định nghĩa	Thể loại phim
	Thuộc tính	<ul style="list-style-type: none"> ❖ id: mã thể loại, mã này là duy nhất ❖ name: tên thể loại
<pre> Participant -profileImageLink: string -name: string -id: string -description: string +addParticipant(): void +updateParticipant(): void +deleteParticipant(): void </pre>	Định nghĩa	Người tham gia vào phim
	Thuộc tính	<ul style="list-style-type: none"> ❖ id: mã người tham gia, mã này là duy nhất ❖ name: tên người tham gia ❖ profileImageLink: url ảnh profile ❖ description: mô tả nhanh về người tham gia
	Phương thức	<ul style="list-style-type: none"> ❖ addParticipant(): thêm người tham gia mới ❖ updateParticipant (): cập nhật người tham gia trong hệ thống ❖ deleteParticipant (): xóa người tham gia khỏi hệ thống

 <pre> class User { -emailAddress: string -phoneNumber: string -roleId: string -birthday: datetime -join_at: datetime -gender: string -bills: Bill[] +updateInfo(): void +addUser(): void +updateUser(): void +deleteUser(): void } </pre>	Định nghĩa	Người dùng hệ thống (khách hàng, nhân viên, quản lý)
	Thuộc tính	<ul style="list-style-type: none"> ❖ emailAddress: địa chỉ email ❖ phoneNumber: số điện thoại ❖ roleId: vai trò trong hệ thống (0 là Admin, 1 là Manager, 2 là Staff, 3 là User) ❖ birthday: ngày sinh ❖ join_at: ngày tham gia ❖ gender: giới tính ❖ bills: danh sách hóa đơn đã mua
 <pre> class Account { -account: string -password: string +addAccount(): void +changePassword(): void +deleteAccount(): void +getAccount(): Account } </pre>	Phương thức	<ul style="list-style-type: none"> ❖ addUser(): thêm người dùng ❖ updateUser(): cập nhật thông tin người dùng ❖ deleteUser(): xóa người dùng ❖ updateInfo(): cập nhật thông tin người dùng
	Định nghĩa	Tài khoản người dùng
 <pre> class Account { -account: string -password: string +addAccount(): void +changePassword(): void +deleteAccount(): void +getAccount(): Account } </pre>	Thuộc tính	<ul style="list-style-type: none"> ❖ account: tên đăng nhập (ở đây là email) ❖ password: mật khẩu tài khoản
	Phương thức	<ul style="list-style-type: none"> ❖ addAccount(): thêm tài khoản vào hệ thống ❖ updateAccount(): cập nhật tài khoản trong hệ thống (đổi mật khẩu) ❖ deleteAccount(): xóa tài khoản khỏi hệ thống ❖ getAccount(): lấy tài khoản bằng tên đăng nhập

<pre> Theater -longitude: float -name: string -id: string -latitude: float -location: string -rooms: Room[] +addTheater(): void +updateTheater(): void +deleteTheater(): void +getTheater(): Theater </pre>	Định nghĩa	Rạp chiếu phim
	Thuộc tính	<ul style="list-style-type: none"> ❖ id: mã rạp, mã này là duy nhất ❖ name: tên rạp ❖ longitude: kinh độ rạp ❖ latitude: vĩ độ rạp ❖ location: địa chỉ rạp ❖ rooms: các phòng chiếu trong rạp
<pre> Room -seats: Seat[] -name: string -id: string -theaterId: string +addRoom(): void +updateRoom(): void +deleteRoom(): void +getRoom(): Room </pre>	Định nghĩa	Phòng chiếu trong rạp
	Thuộc tính	<ul style="list-style-type: none"> ❖ id: mã phòng chiếu, mã này là duy nhất ❖ name: tên phòng chiếu ❖ theaterId: mã rạp chiếu ❖ seats: danh sách ghế trong phòng
<pre> Seat -seatTypeID: string -name: string -id: string -seatCor: string +updateSeatType() </pre>	Định nghĩa	Ghế ngồi trong phòng chiếu
	Thuộc tính	<ul style="list-style-type: none"> ❖ id: mã ghế, mã này là duy nhất ❖ name: tên ghế ❖ seatCor: vị trí ghế trong phòng chiếu (A1, B10...) ❖ seatTypeID: mã thể loại ghế
	Phương thức	<ul style="list-style-type: none"> ❖ updateSeat(): cập nhật ghế trong phòng chiếu (cập nhật loại ghế)

<pre> Product -<i>id</i>: string +<i>getPrice()</i>: int +<i>add()</i>: void +<i>update()</i>: void +<i>delete()</i>: void </pre>	Định nghĩa	Sản phẩm bán trong rạp chiếu phim (vé xem phim, đồ ăn)
	Thuộc tính	❖ <i>id</i> : mã sản phẩm, mã này là duy nhất
	Phương thức	❖ <i>add()</i> : thêm sản phẩm mới vào hệ thống ❖ <i>update()</i> : cập nhật sản phẩm trong hệ thống ❖ <i>delete()</i> : xóa sản phẩm khỏi hệ thống ❖ <i>getPrice()</i> : lấy giá của sản phẩm
<pre> Ticket -<i>schedule</i>: Schedule -<i>ordered</i>: bool -<i>seat</i>: Seat +<i>order()</i>: void </pre>	Định nghĩa	Vé xem phim
	Thuộc tính	❖ <i>scheduleId</i> : mã lịch chiếu ❖ <i>ordered</i> : trạng thái vé (true: đã đặt, false: chưa đặt) ❖ <i>seat</i> : chỗ ngồi đã đặt
	Phương thức	❖ <i>order()</i> : đặt vé
<pre> Food -<i>name</i>: String -<i>price</i>: int -<i>linkImg</i>: String +<i>updatePrice()</i>: void +<i>updateImgLink()</i>: void </pre>	Định nghĩa	Đồ ăn
	Thuộc tính	❖ <i>price</i> : giá đồ ăn ❖ <i>name</i> : tên đồ ăn ❖ <i>linkImg</i> : url ảnh minh họa
	Phương thức	❖ <i>updatePrice()</i> : cập nhật giá đồ ăn trong hệ thống ❖ <i>updateImgLink()</i> : cập nhật ảnh minh họa đồ ăn trong hệ thống
<pre> Stream_schedule -<i>movie</i>: Movie -<i>id</i>: string -<i>roomID</i>: string -<i>startShowTime</i>: int -<i>seatPrice</i>: Dictionary<Seat, int> -<i>showTimeDay</i>: date +<i>deleteSchedule()</i>: void +<i>updateSchedule()</i>: void +<i>addSchedule()</i>: void +<i>getSchedule()</i>: Schedule </pre>	Định nghĩa	Lịch chiếu phim
	Thuộc tính	❖ <i>id</i> : mã lịch chiếu, mã này là duy nhất ❖ <i>movie</i> : phim được chiếu ❖ <i>roomID</i> : mã phòng chiếu ❖ <i>startShowTime</i> : giờ chiếu (phút) ❖ <i>seatPrice</i> : giá vé dựa vào thể loại ghế ❖ <i>showTimeDay</i> : ngày chiếu
	Phương thức	❖ <i>addSchedule()</i> : thêm lịch chiếu mới vào hệ thống ❖ <i>updateSchedule()</i> : cập nhật lịch chiếu trong hệ thống ❖ <i>deleteSchedule()</i> : xóa lịch chiếu khỏi hệ thống ❖ <i>getSchedule()</i> : tìm kiếm lịch chiếu dựa vào các tiêu chí (phim, ngày, giờ...)

<pre> Bill -id: string -customerID: string -voucher: Voucher -date: date -paymentMethodId: string +addBill(): void +useVoucher(): void +getTotalPrice(): double </pre>	Định nghĩa	Hóa đơn mua hàng tại rạp chiếu phim
	Thuộc tính	<ul style="list-style-type: none"> ❖ id: mã hóa đơn, mã này là duy nhất ❖ customerID: mã khách hàng của hóa đơn ❖ voucher: mã giảm giá áp dụng vào hóa đơn ❖ date: ngày giờ hóa đơn được tạo ❖ paymentMethodID: mã phương thức thanh toán
<pre> Bill_detail -productId: string -billID: string -quantity: unsigned int +updateQuantity(): void +getProduct(): Product </pre>	Định nghĩa	Chi tiết hóa đơn
	Thuộc tính	<ul style="list-style-type: none"> ❖ productId: mã sản phẩm đã mua trong hóa đơn ❖ billID: mã hóa đơn ❖ quantity: số lượng mua
<pre> Voucher -code: string -description: string -discountRate: int -discountType: int +addVoucher(): void +updateVoucher(): void +deleteVoucher(): void +checkExist(): bool </pre>	Định nghĩa	Mã giảm giá
	Thuộc tính	<ul style="list-style-type: none"> ❖ code: mã thẻ loại, mã này là duy nhất ❖ description: thẻ lệ giảm giá ❖ discountRate: tỉ lệ giảm giá ❖ discountType: loại giảm giá (theo %, theo giá cụ thể....)
	Phương thức	<ul style="list-style-type: none"> ❖ addGenre(): thêm mã giảm giá mới vào hệ thống ❖ updateGenre(): cập nhật mã giảm giá trong hệ thống ❖ deleteGenre(): xóa mã giảm giá khỏi hệ thống ❖ checkExist(): kiểm tra sự tồn tại của mã giảm giá

<table border="1"> <tr><td>Voucher_User</td></tr> <tr><td>-voucher: Voucher</td></tr> <tr><td>-customer: Customer</td></tr> <tr><td>-applied: bool</td></tr> <tr><td>+reset(): bool</td></tr> <tr><td>+apply(): bool</td></tr> <tr><td>+checkUsed(): bool</td></tr> </table>	Voucher_User	-voucher: Voucher	-customer: Customer	-applied: bool	+reset(): bool	+apply(): bool	+checkUsed(): bool	Định nghĩa	Liên kết mã giảm giá với người dùng, để kiểm tra xem người dùng đã dùng mã giảm giá
Voucher_User									
-voucher: Voucher									
-customer: Customer									
-applied: bool									
+reset(): bool									
+apply(): bool									
+checkUsed(): bool									
Thuộc tính	<ul style="list-style-type: none"> ❖ voucher: mã giảm giá ❖ customer: người dùng ❖ applied: trạng thái sử dụng (true: đã sử dụng, false: chưa sử dụng) 								
Phương thức	<ul style="list-style-type: none"> ❖ reset(): đưa trạng thái mã về chưa sử dụng ❖ apply(): sử dụng mã giảm giá ❖ checkUsed(): kiểm tra đã sử dụng hay chưa 								
<table border="1"> <tr><td>PaymentMethod</td></tr> <tr><td>-id: string</td></tr> <tr><td>-name: string</td></tr> <tr><td>-icon_link: string</td></tr> <tr><td>+add(): void</td></tr> <tr><td>+delete(): void</td></tr> <tr><td>+update(): void</td></tr> </table>	PaymentMethod	-id: string	-name: string	-icon_link: string	+add(): void	+delete(): void	+update(): void	Định nghĩa	Phương thức thanh toán
PaymentMethod									
-id: string									
-name: string									
-icon_link: string									
+add(): void									
+delete(): void									
+update(): void									
Thuộc tính	<ul style="list-style-type: none"> ❖ id: mã phương thức thanh toán, mã này là duy nhất ❖ name: tên phương thức thanh toán ❖ icon_link: url đến icon đại diện phương thức thanh toán 								
Phương thức	<ul style="list-style-type: none"> ❖ add(): thêm phương thức thanh toán ❖ delete(): xóa phương thức thanh toán ❖ update(): cập nhật phương thức thanh toán 								

Phần 4. Pha thiết kế

4.1 Lựa chọn công nghệ mạng

Trong báo cáo này, em sẽ sử dụng kiến trúc hai tầng client-server để xây dựng ứng dụng mobile.

Với kiến trúc hai tầng, chương trình và dữ liệu phải được chuyển từ máy tính trung tâm sang máy khách. Điều này đòi hỏi phải có sự kết nối nhanh giữa hai bên và điều này dẫn đến ý tưởng hiện đại về mạng.

Kiến trúc hai tầng (two tier architecture) là thế hệ kiến trúc tiếp theo được phổ biến vào những năm 1970. Ý tưởng của kiến trúc này là nhằm tăng cường khả năng xử lý trên mỗi máy khách để cho các máy tính trung tâm không phải thực hiện tất cả các tiến trình.

4.1.1 Hoạt động của mô hình client-server

Máy khách (Client) và máy chủ (Server) trao đổi thông tin với nhau dưới dạng các thông điệp (Message). Thông điệp gửi từ máy khách sang máy chủ gọi là các thông điệp yêu cầu (Request Message) mô tả công việc mà phần mềm máy khách muốn máy chủ thực hiện.

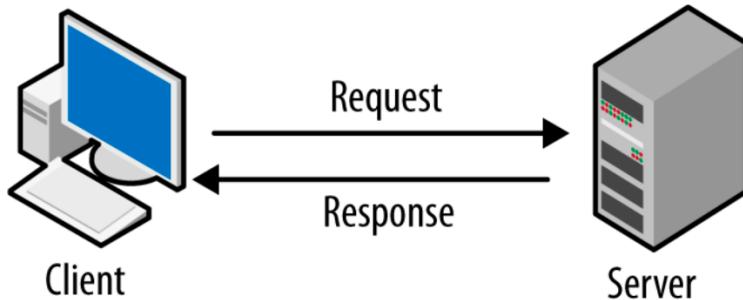


Figure 67 Mô hình client-server

Mỗi khi máy chủ nhận được một thông điệp yêu cầu, máy chủ sẽ phân tích yêu cầu, thực thi công việc theo yêu cầu và gửi kết quả về máy khách (nếu có) trong một thông điệp trả lời (Reply Message). Khi vận hành, một máy chủ sẽ phục vụ cho nhiều máy khách.

Thông thường phần mềm máy khách đảm nhận các chức năng về giao diện người dùng, như tạo các form nhập liệu, các thông báo, các biểu báo giao tiếp với người dùng; phần mềm máy chủ này đảm nhận các chức năng về xử lý và lưu trữ dữ liệu. Mô hình tạo điều kiện dễ dàng cho việc bảo trì, chia sẻ, tổng hợp dữ liệu trong toàn bộ công ty hoặc tổ chức. [1]

4.1.2 Ưu điểm của mô hình

- Giúp chúng ta có thể làm việc trên bất kỳ một máy tính nào có hỗ trợ giao thức truyền thông. Giao thức chuẩn này cũng giúp các nhà sản xuất tích hợp lên nhiều sản phẩm khác nhau mà không gặp phải khó khăn gì.
- Có thể có nhiều server cùng làm một dịch vụ, chúng có thể nằm trên nhiều máy tính hoặc một máy tính.
- Chỉ mang đặc điểm của phần mềm mà không hề liên quan đến phần cứng, ngoài yêu cầu duy nhất là server phải có cấu hình cao hơn các client.
- Hỗ trợ người dùng nhiều dịch vụ đa dạng và sự tiện dụng bởi khả năng truy cập từ xa.
- Cung cấp một nền tảng lý tưởng, cho phép cung cấp tích hợp các kỹ thuật hiện đại như mô hình thiết kế hướng đối tượng, hệ chuyên gia, hệ thông tin địa lý (GIS).

4.1.3 Nhược điểm của mô hình

Vấn đề bảo mật dữ liệu thông tin đôi khi còn chưa được an toàn lắm. Vì do phải trao đổi dữ liệu giữa 2 máy tính khác nhau ở 2 khu vực địa lý cách xa nhau. Và đây cũng là nhược điểm duy nhất của mô hình này. Tuy nhiên vấn đề này thì có một số giao thức đã hỗ trợ bảo mật dữ liệu khi truyền tải. Giao thức được sử dụng phổ biến như HTTPS.

4.2 Xây dựng kiến trúc

Phát triển một ứng dụng phần mềm bằng cách áp dụng một mẫu kiến trúc phần mềm luôn được các nhà phát triển ưu tiên. Một mẫu kiến trúc mang lại tính module cho các tệp dự án và giúp cấu trúc code bớt phức tạp, dễ hiểu hơn. MVC (Model - View - Controller) và MVVM (Model - View - ViewModel) là hai kiến trúc phần mềm phổ biến nhất trong số đó. Hai kiến trúc này có khá nhiều điểm tương đồng, nhưng MVVM tỏ ra hiệu quả hơn khi áp dụng vào những ứng dụng mobile, đồng thời MVVM cũng có một số lợi thế so với MVC.

4.2.1 Tổng quan về kiến trúc MVC

Mẫu MVC chia cấu trúc dự án thành 3 phần: Model, View và Controller. Khi tạo các file/class trong dự án, lập trình viên phải phân loại nó thành 1 trong 3 phần sau:

- **Model:** Thành phần này chịu trách nhiệm lưu trữ dữ liệu của ứng dụng. Lớp Model không cần quan tâm hay xử lý gì với lớp View. Lớp này xử lý các logic nghiệp vụ và giao tiếp với các lớp cơ sở dữ liệu và mạng.
- **View:** Đây là lớp giao diện người dùng (UI) chứa các thành phần hiển thị trên màn hình. Hơn nữa, lớp này chịu trách nhiệm trực quan hóa các dữ liệu thô thành giao diện đẹp mắt và nhận các tương tác của người dùng.
- **Controller:** Thành phần này thiết lập mối quan hệ giữa View và Model. Lớp này chứa logic ứng dụng cốt lõi và nhận thông tin phản hồi từ người dùng, sau đó cập nhật Model theo nhu cầu.

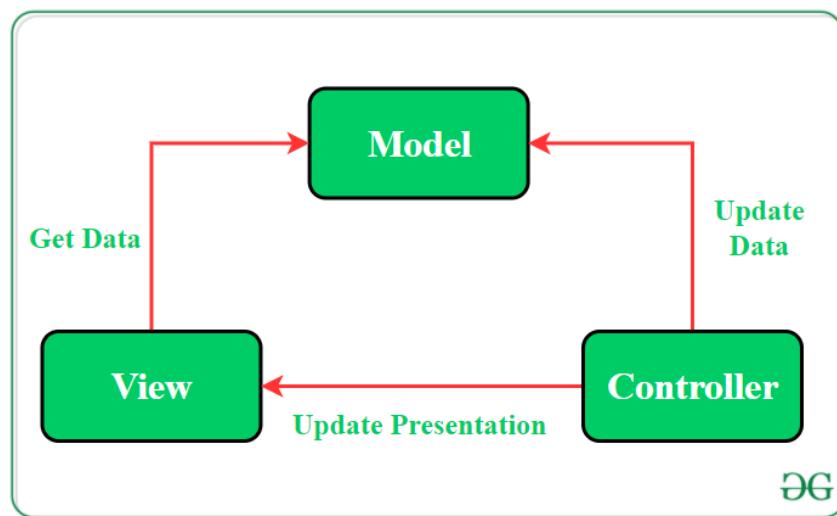


Figure 68 Cấu trúc của mẫu MVC [2]

4.2.2 Tổng quan về kiến trúc MVVM

MVVM đề xuất tách biệt logic trình bày dữ liệu (Views hoặc UI) khỏi phần logic nghiệp vụ cốt lõi của ứng dụng. Các lớp của MVVM:

- **Model:** Lớp này chịu trách nhiệm trừu tượng hóa dữ liệu. Model và ViewModel phối hợp với nhau để lấy và lưu dữ liệu.

- **View:** Mục đích của lớp này là thông báo cho ViewModel về hành động của người dùng. Lớp này quan sát ViewModel và không chứa bất kỳ loại logic ứng dụng nào.
- **ViewModel:** Lớp này lấy dữ liệu qua lớp Model và cung cấp cho lớp View, nó đóng vai trò là cầu nối giữa Model và View.

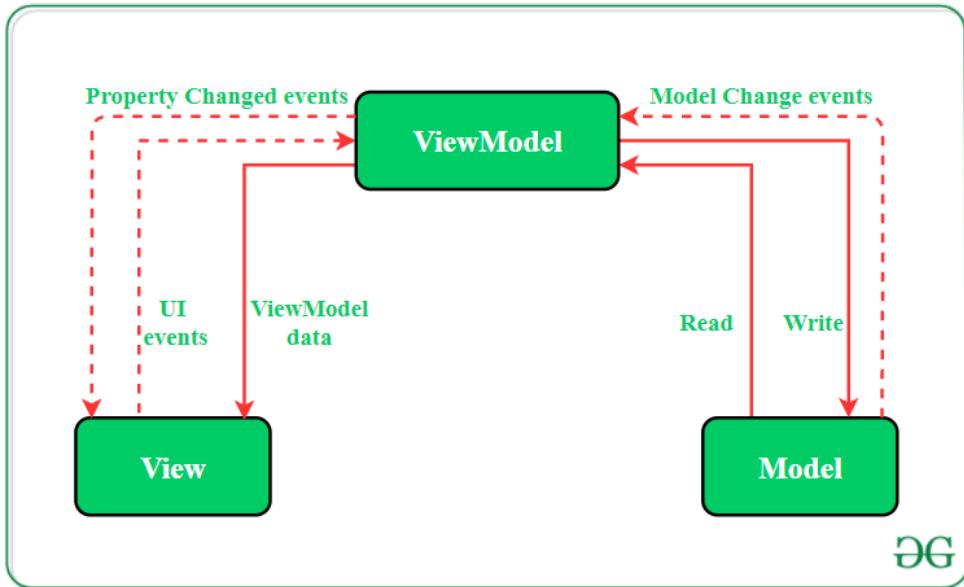


Figure 69 Cấu trúc mẫu MVVM [2]

4.2.3 Sự khác nhau giữa MVC và MVVM

MVC	MVVM
Mẫu kiến trúc lâu đời nhất dành trong ngành phát triển phần mềm.	Mẫu kiến trúc được công nhận trong ngành cho các ứng dụng.
Tương tác đầu vào của người dùng được xử lý bởi lớp Controller	Lớp View nhận đầu vào từ người dùng và chuyển tiếp sang lớp ViewModel
Controller và View tồn tại với mối quan hệ một - nhiều. Một Controller có thể chọn View khác nhau dựa trên hoạt động yêu cầu.	Nhiều View có thể được ánh xạ với một ViewModel và do đó, tồn tại mối quan hệ một-nhiều giữa View và ViewModel.
Lớp View không biết đến sự tồn tại của lớp Controller.	Lớp View có tham chiếu đến lớp ViewModel.
Kiến trúc này phụ thuộc nhiều vào các API của Android.	Có ít hoặc không phụ thuộc vào các API của Android.
Khó cập nhật và bảo trì các tính năng của ứng dụng vì các lớp phụ thuộc mạnh.	Dễ dàng cập nhật và bảo trì. Tuy nhiên nếu logic liên kết dữ liệu quá phức tạp sẽ dẫn đến khó khăn trong debug.

Hỗ trợ hạn chế cho unit test.	Khả năng kiểm thử unit test cao.
Không tuân theo nguyên tắc module và nguyên tắc single responsibility.	Tuân theo nguyên tắc module và nguyên tắc single responsibility.

4.2.4 Lựa chọn kiến trúc

Nhìn chung, sự khác biệt chính giữa MVC và MVVM là vai trò của thành phần trung gian. Trong MVC, Controller đóng vai trò trung gian giữa View và Model, trong khi đó trong MVVM, ViewModel đóng vai trò trung gian giữa View và Model. MVC là một mẫu đơn giản hơn, trong khi MVVM linh hoạt hơn và cho phép phân tách rõ ràng hơn giữa các lớp khác nhau của ứng dụng. Mặc dù cả MVC và MVVM đều là các mẫu hữu ích để tổ chức mã cho một ứng dụng phần mềm, MVVM mang lại nhiều lợi ích hơn so với MVC, dưới đây là một số điểm lợi ích của MVVM:

- **Tách biệt rõ ràng các lớp nhiệm vụ:** Trong MVVM, ViewModel đóng vai trò là trung gian giữa View và Model, điều này cho phép phân tách rõ ràng hơn giữa các lớp khác nhau của ứng dụng. Điều này có thể làm cho việc duy trì và sửa đổi mã nguồn trở nên dễ dàng hơn theo thời gian.
- **Khả năng kiểm thử tốt hơn:** Vì ViewModel chịu trách nhiệm lấy dữ liệu từ Model chuyển tiếp cho View, việc kiểm thử ViewModel riêng biệt so với View có thể dễ dàng hơn. Điều này có thể làm cho việc đảm bảo ứng dụng hoạt động chính xác và dễ dàng xác định và sửa lỗi trở nên thuận tiện hơn.
- **Độ linh hoạt cao hơn:** ViewModel trong MVVM linh hoạt hơn Model trong MVC, vì nó có thể lấy dữ liệu và logic từ Model một cách dễ dàng hơn để chuyển tiếp cho View. Điều này có thể làm cho việc tạo ra các giao diện khác nhau cho cùng dữ liệu và logic hoặc tái sử dụng cùng một ViewModel với nhiều View trở nên dễ dàng hơn.
- **Hỗ trợ tốt hơn cho nhiều nhà phát triển:** Bởi vì mẫu MVVM cho phép phân tách rõ ràng các quan tâm giữa các lớp khác nhau của ứng dụng, việc nhiều nhà phát triển làm việc trên các phần khác nhau của mã nguồn đồng thời có thể dễ dàng hơn. Điều này có thể cải thiện hiệu quả quá trình phát triển và giảm thiểu rủi ro xung đột giữa các phần khác nhau của mã nguồn.

Những lợi ích này làm cho mẫu MVVM trở thành một lựa chọn hữu ích và hiệu quả hơn so với MVC trong việc phát triển ứng dụng, đặc biệt là trong các dự án có quy mô lớn và phức tạp. Chính vì vậy, em lựa chọn MVVM để phát triển ứng dụng trong đồ án này.

4.3 Thiết kế tương tranh và an toàn, bảo mật

4.3.1 Mã hóa thông tin nhạy cảm

Để đảm bảo tính bảo mật khi hacker bằng một cách nào đó lấy được thông tin người dùng (chẳng hạn như chặn được dữ liệu truyền đi) thì cũng không biết được mật khẩu là gì, em sẽ băm mật khẩu trước khi gửi lên server. Cụ thể thì trong báo cáo này, em sử dụng hàm băm SCRYPT.

Hàm băm chuyển đổi mật khẩu gốc thành một chuỗi ký tự dài cố định, thường là không thể đảo ngược. Điều này có nghĩa là ngay cả khi kẻ tấn công có được giá trị băm, họ cũng không thể dễ dàng chuyển ngược lại để tìm ra mật khẩu gốc.

Vậy nhưng kể cả khi đã dùng hàm băm, hacker vẫn có thể sử dụng rainbow table để lấy dữ liệu thông qua hash collision.

Rainbow table là một bảng tính toán trước dùng để đảo ngược các chuỗi mã hóa về ngược lại giá trị ban đầu. Các chuỗi trong bảng được tạo bằng cách lặp đi lặp lại việc áp dụng hashing cho các mật khẩu gốc, sau đó sử dụng reduction function và tiếp tục hash. Lặp lại đến khi chúng ta có cặp (password, hash) - hash ở đây là chuỗi cuối cùng vòng lặp.

Vì vậy, em đã thêm một số tham số vào hàm băm:

- Salt: Là dữ liệu ngẫu nhiên được đưa vào như một đầu vào bổ sung cho hàm băm dùng để băm dữ liệu. Việc sử dụng salt giúp bảo vệ chống lại các cuộc tấn công sử dụng các bảng tính toán trước (ví dụ như rainbow tables), bằng cách mở rộng kích thước của bảng cần thiết một cách đáng kể. Salt vẫn có thể bị hack tuy nhiên thay vì bạn kiểm trên 1 dictionary có 100 trang, thì giờ nó là 1000000 trang và từ 1 tuần, nó sẽ biến thành 1 năm!
- Hash_rounds: Đề cập đến số lần một hàm băm được áp dụng lặp đi lặp lại trên dữ liệu đầu vào. Đây là một kỹ thuật được sử dụng để tăng cường bảo mật của quá trình băm mật khẩu. Việc tăng số vòng băm làm cho việc tính toán hàm băm tốn nhiều thời gian hơn, từ đó làm chậm quá trình tấn công brute-force (thử tất cả các mật khẩu có thể) hoặc tấn công sử dụng các bảng tính toán trước.

Cụ thể thì trong báo cáo này, em sử dụng hàm băm SCRYPT cùng với salt, số vòng băm là 8, và sử dụng cả chữ ký số để đảm bảo tính chính xác của mật khẩu được lưu.

```
hash_config {
    algorithm: SCRYPT,
    base64_signer_key: [REDACTED]
    base64_salt_separator: [REDACTED]
    rounds: [REDACTED]
    mem_cost: [REDACTED]
}
```

Figure 70 Hash config

4.3.2 Phòng chống Brute Force

Tấn công brute-force là một phương pháp bẻ khóa phổ biến. Một cuộc tấn công brute-force liên quan đến việc 'đoán' tên người dùng và mật khẩu để truy cập trái phép vào hệ thống, hacker sẽ sử dụng phương pháp thử và sai để cố gắng đoán thông tin đăng nhập hợp lệ. Ngoài ra ta có thể sử dụng brute-force để khai thác OTP, Timestamp, Cookie, vv...

Brute-force là một phương pháp tấn công đơn giản và có tỷ lệ thành công cao. Bởi vì tùy thuộc vào độ dài và độ phức tạp của mật khẩu, việc bẻ khóa mật khẩu có thể mất từ vài giây đến nhiều năm. Do đó các trang web sử dụng phương thức đăng nhập dựa trên mật khẩu hoàn toàn có thể rất dễ bị tấn công nếu họ không thực hiện đầy đủ biện pháp bảo vệ.

Để hạn chế tấn công brute-force, có nhiều biện pháp thực hiện. Cụ thể trong demo của mình, em đã thực hiện 2 phương pháp sau:

- Yêu cầu mật khẩu mạnh: yêu cầu tối thiểu cho mật khẩu của người dùng là phải đạt ít nhất độ dài bằng 8 và chứa cả chữ cái lẫn chữ số
- Hạn chế số lần đăng nhập sai: Giới hạn số lần thử cũng làm giảm khả năng bị tấn công brute-force. Đi kèm với đó là việc làm tăng thời gian cho phép nhập khi nhập quá nhiều lần sai. Cụ thể trong chương trình demo, em lựa chọn đăng nhập với email được built-in với Firebase, sẽ tự động giới hạn đăng nhập với địa chỉ IP request quá nhiều lần trong 1 khoảng thời gian.

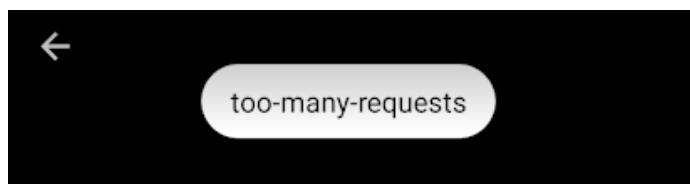


Figure 71 Giới hạn đăng nhập khi request nhiều lần trong thời gian ngắn

4.4 Thiết kế logic

4.4.1 Thiết kế app client (mobile)

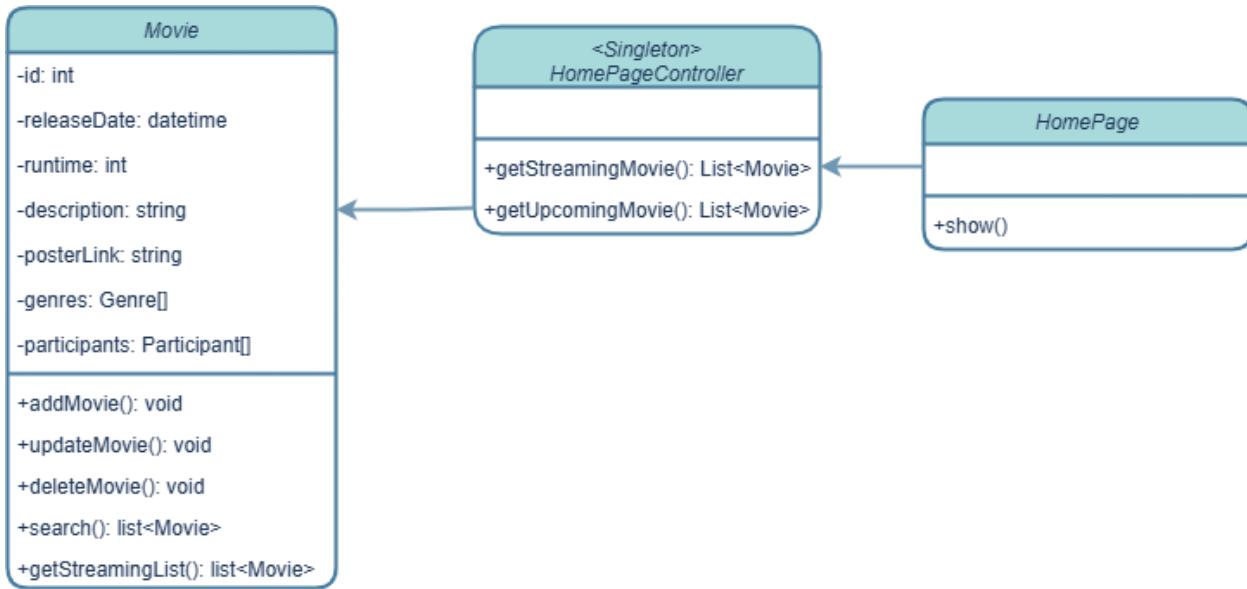


Figure 72 Design class cho chức năng hiển thị trang chủ

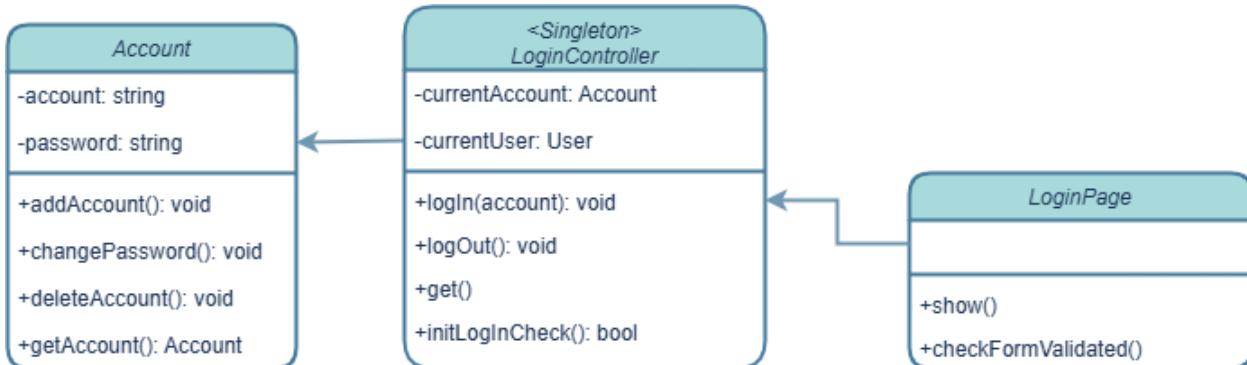


Figure 73 Design class cho chức năng đăng nhập

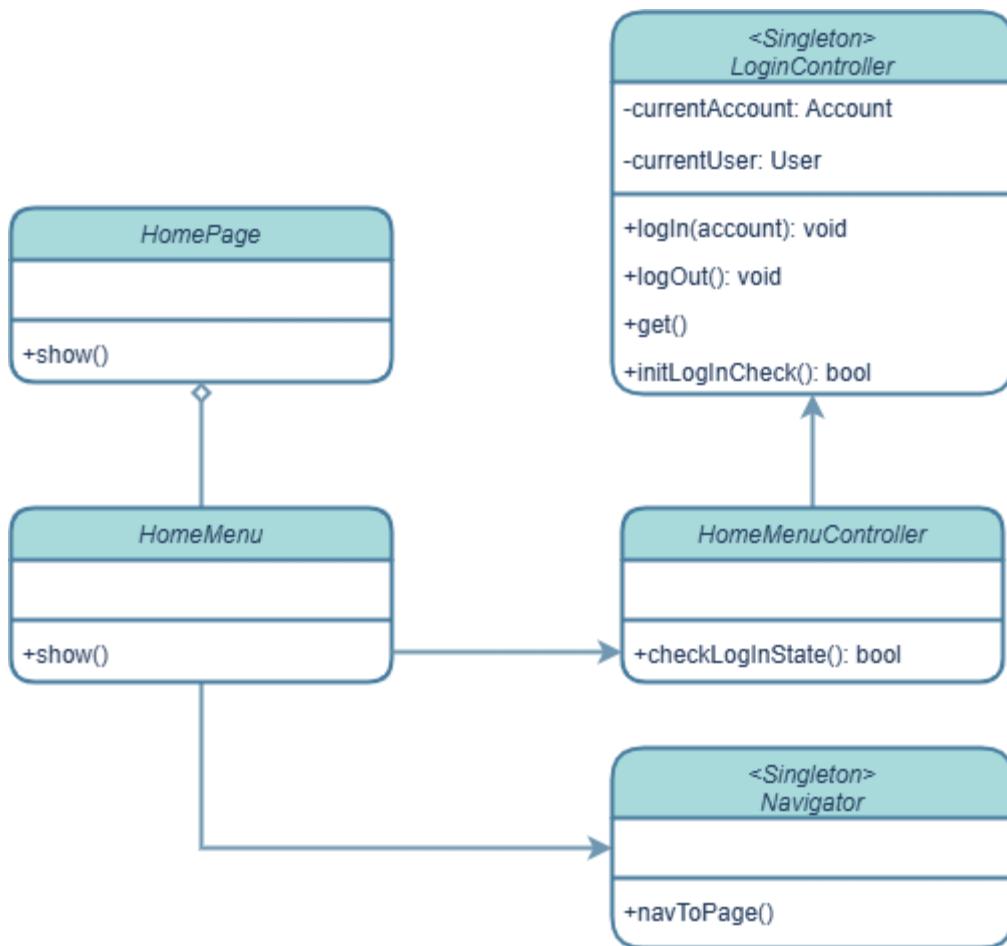


Figure 74 Design class cho chức năng hiển thị menu

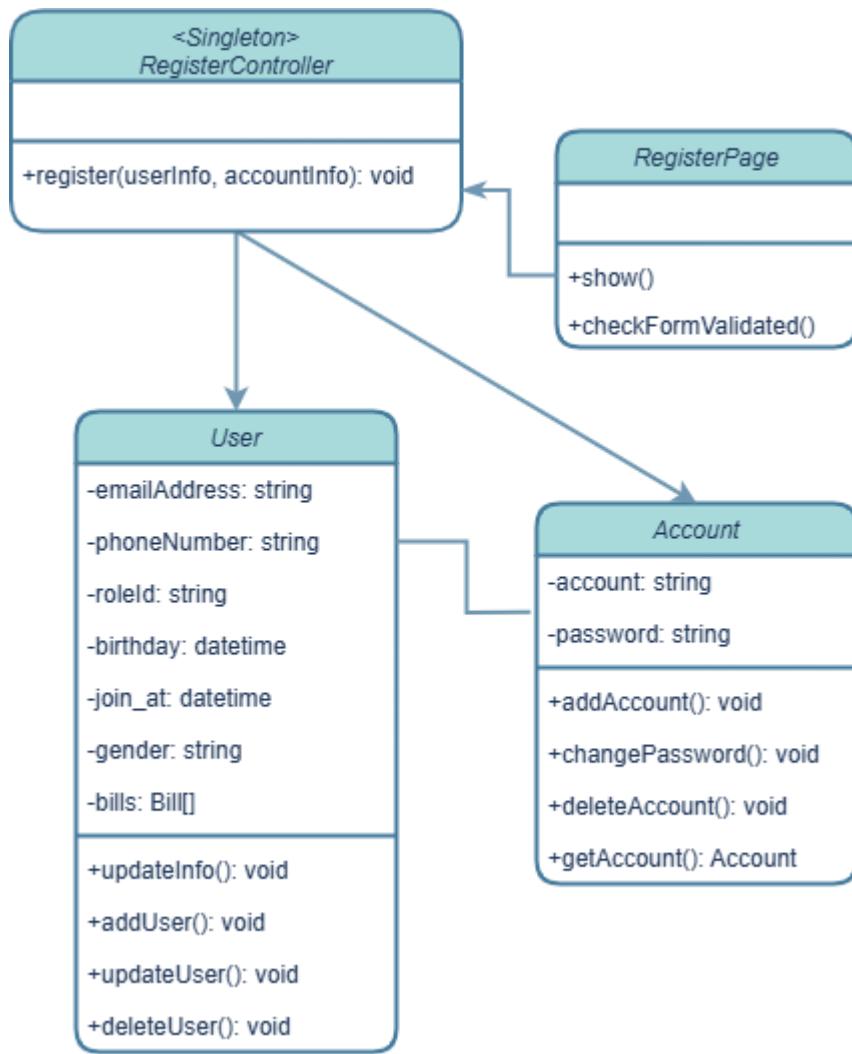


Figure 75 Design class cho chức năng đăng ký

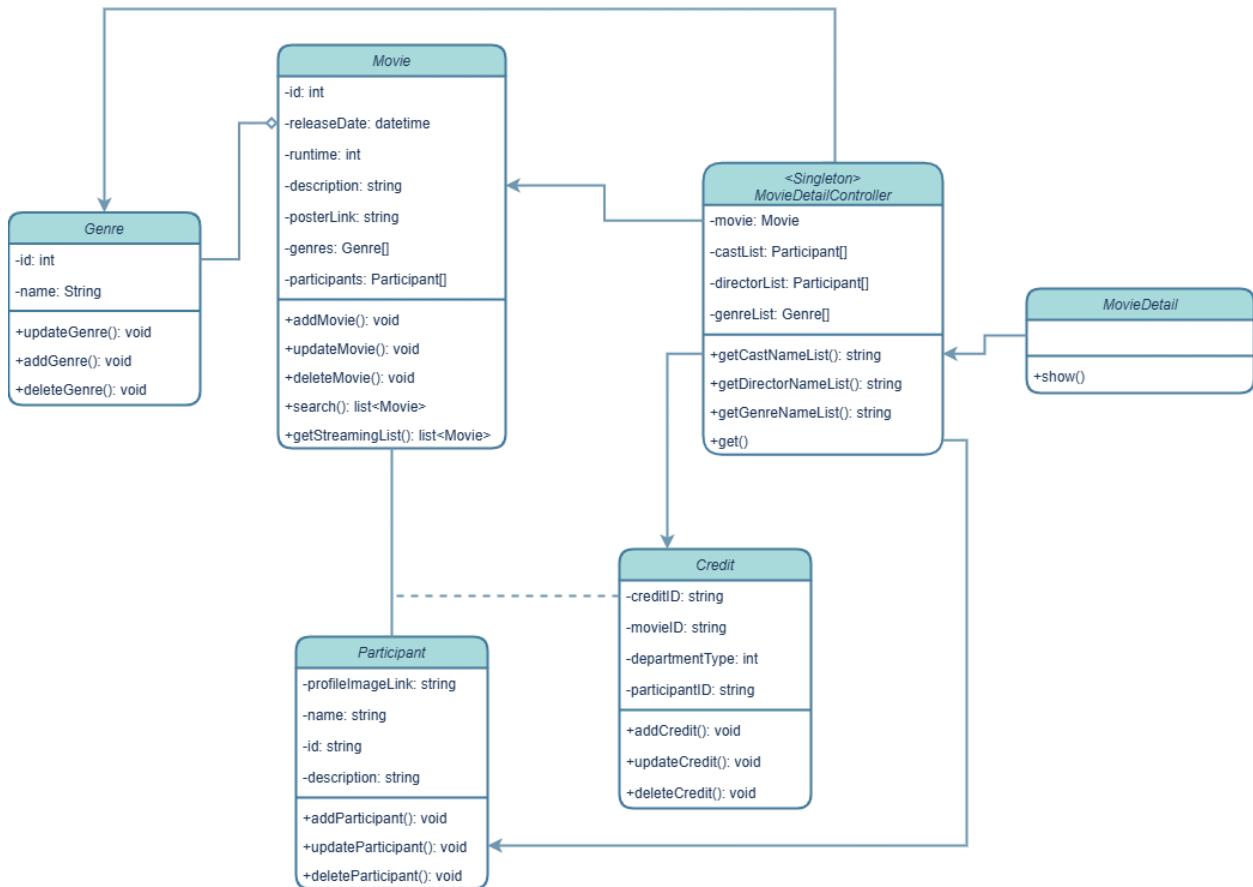


Figure 76 Design class cho chức năng xem chi tiết phim

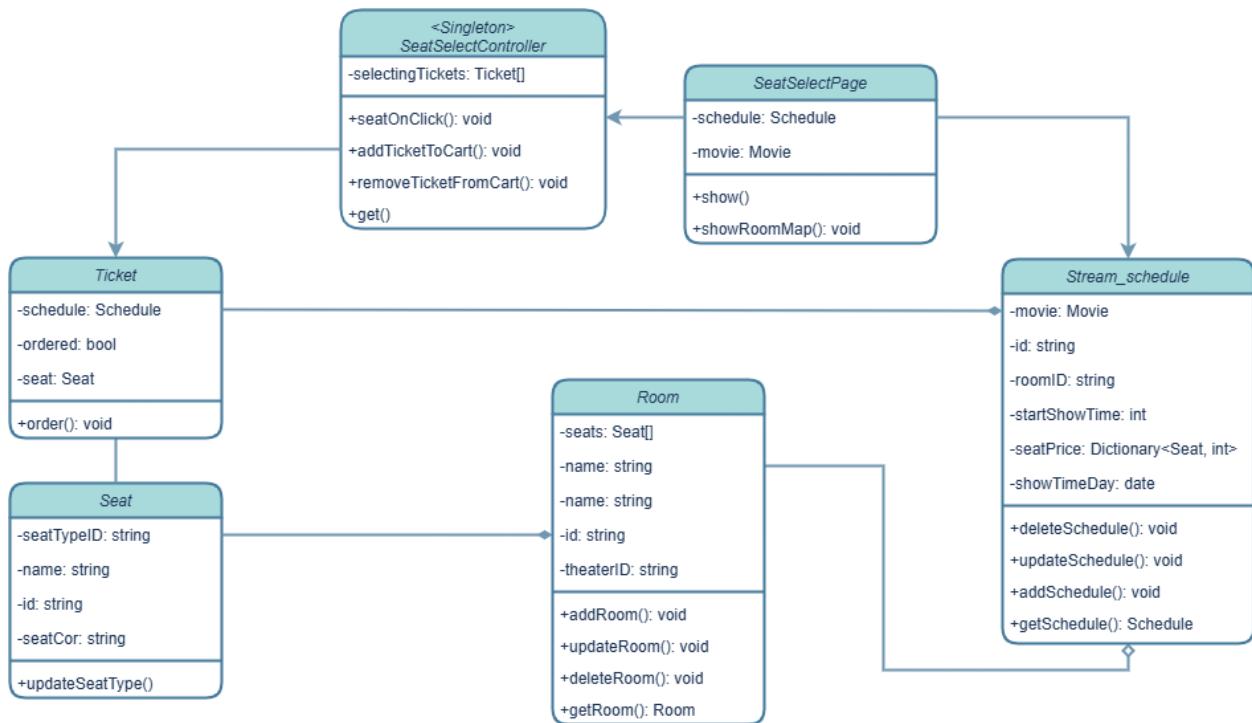


Figure 77 Design class cho chức năng chọn ghế

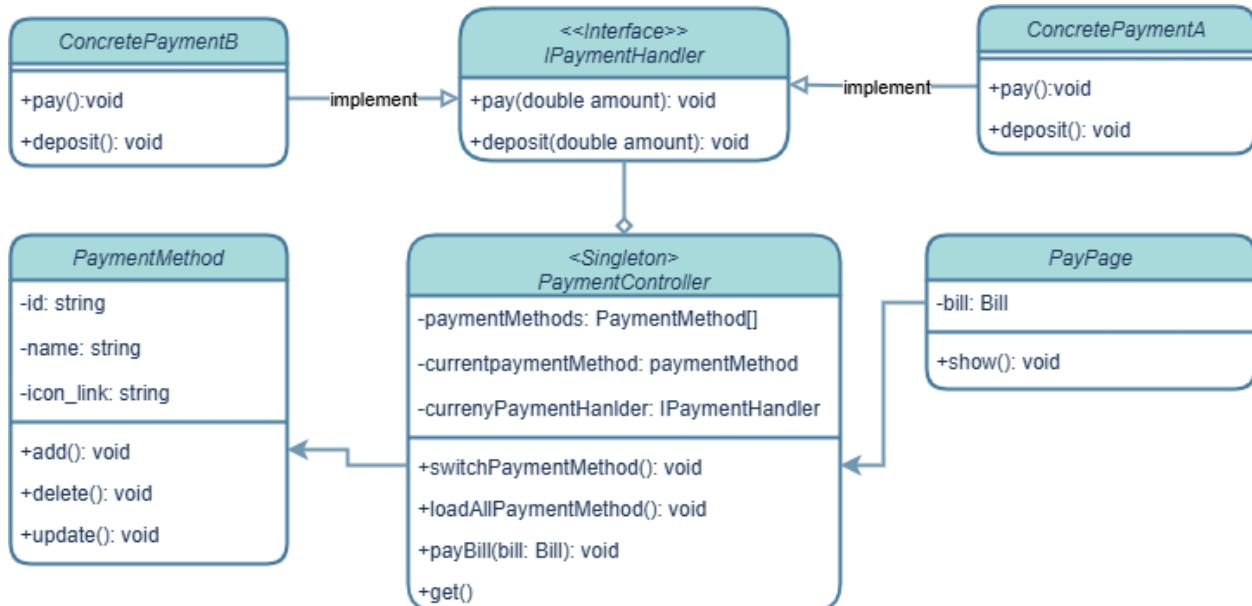


Figure 78 Design class cho chức năng thanh toán

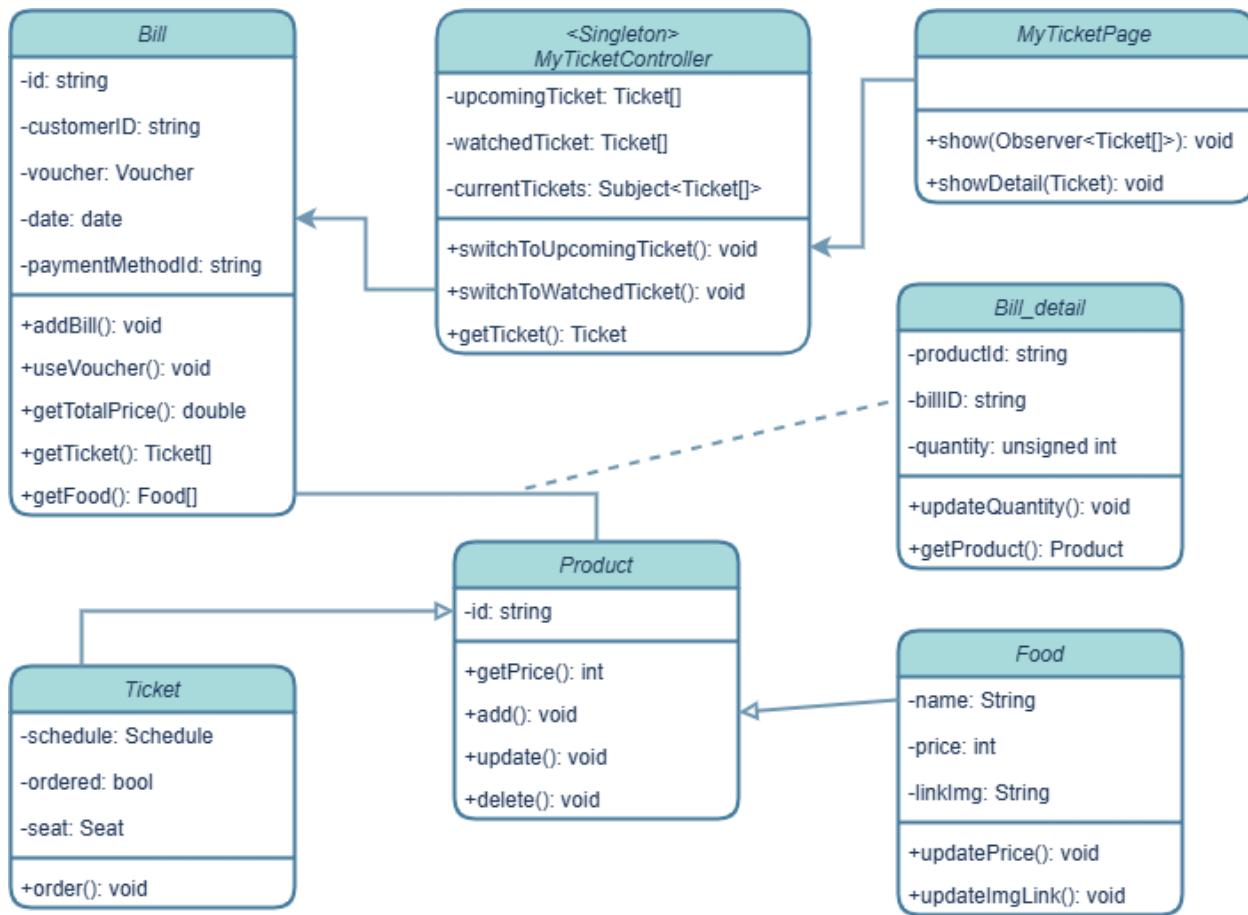


Figure 79 Design class cho chức năng xem thông tin hóa đơn

4.4.2 Thiết kế app quản trị (window)

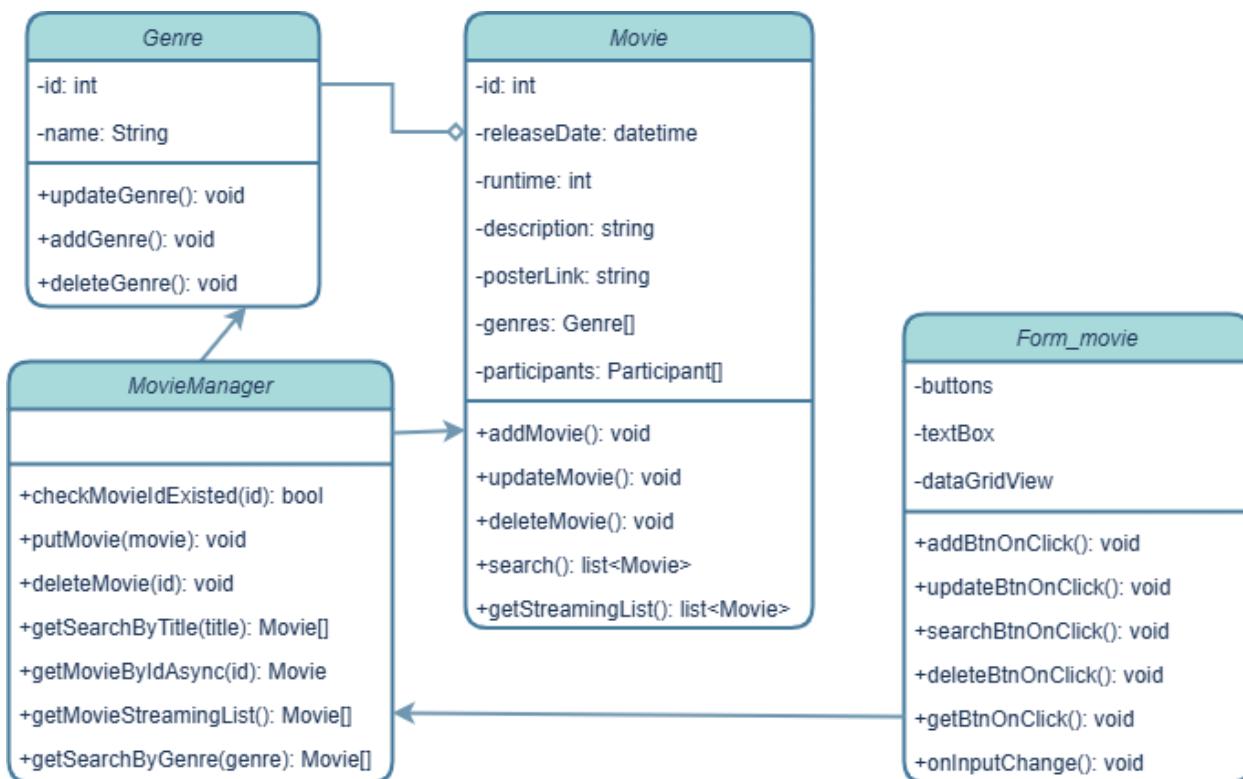


Figure 80 Thiết kế logic form quản lý phim

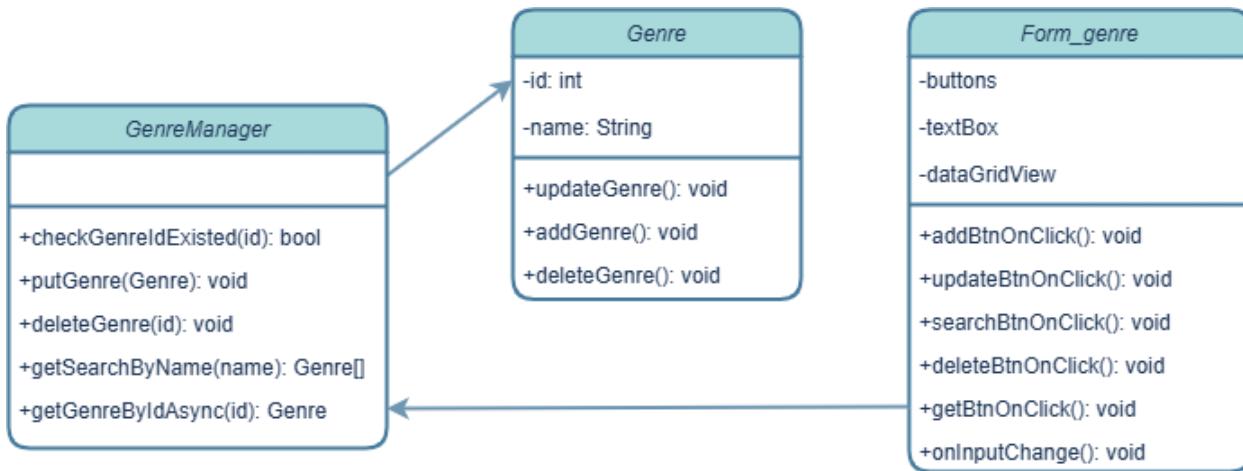


Figure 81 Thiết kế logic form quản lý thể loại phim

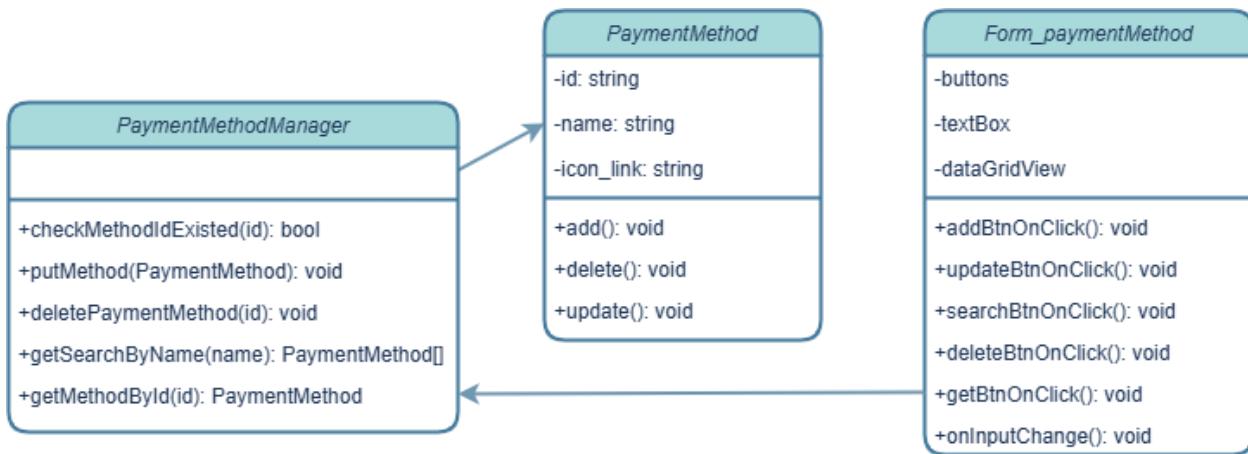


Figure 82 Thiết kế logic form quản lý phương thức thanh toán

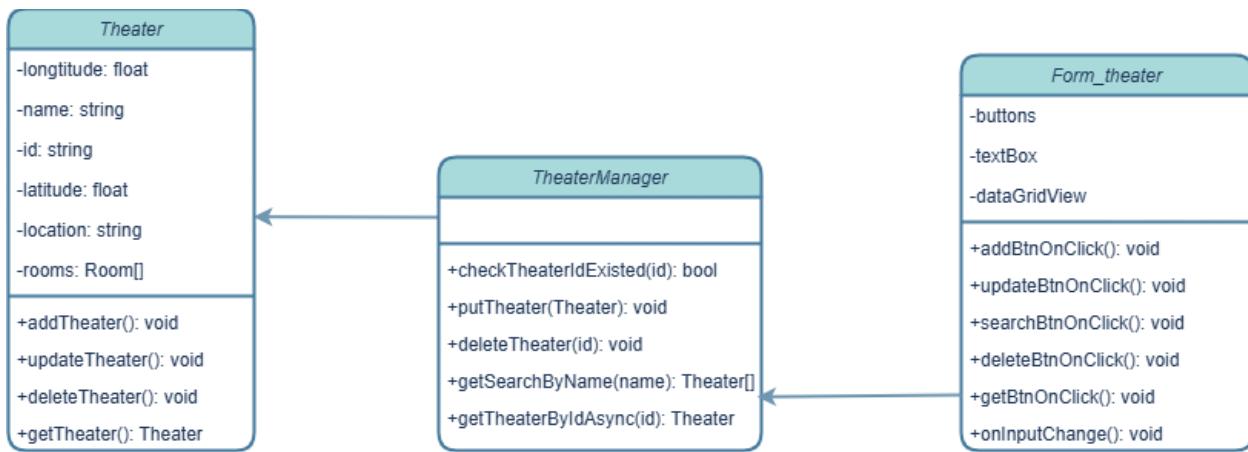


Figure 83 Thiết kế logic form quản lý rạp chiếu

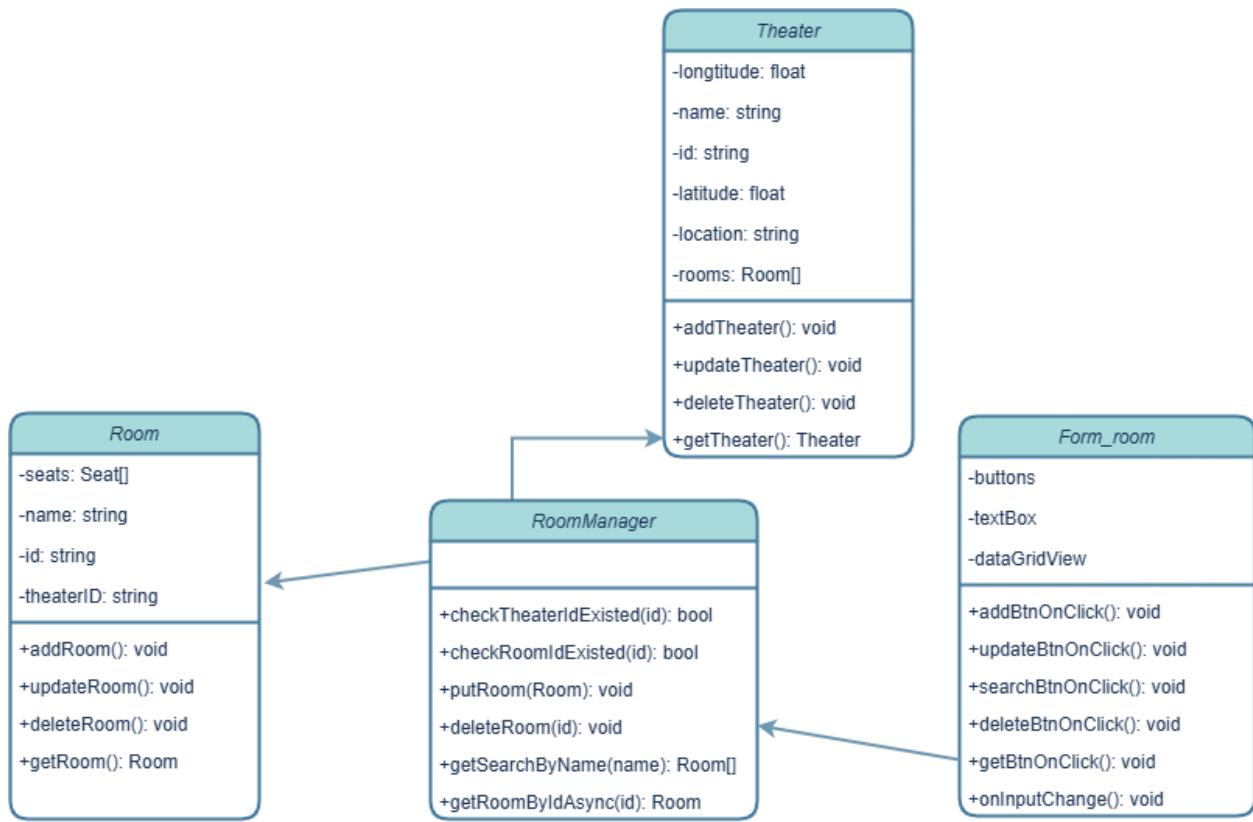


Figure 84 Thiết kế logic form quản lý phòng chiếu



Figure 85 Thiết kế logic form quản lý sơ đồ phòng chiếu

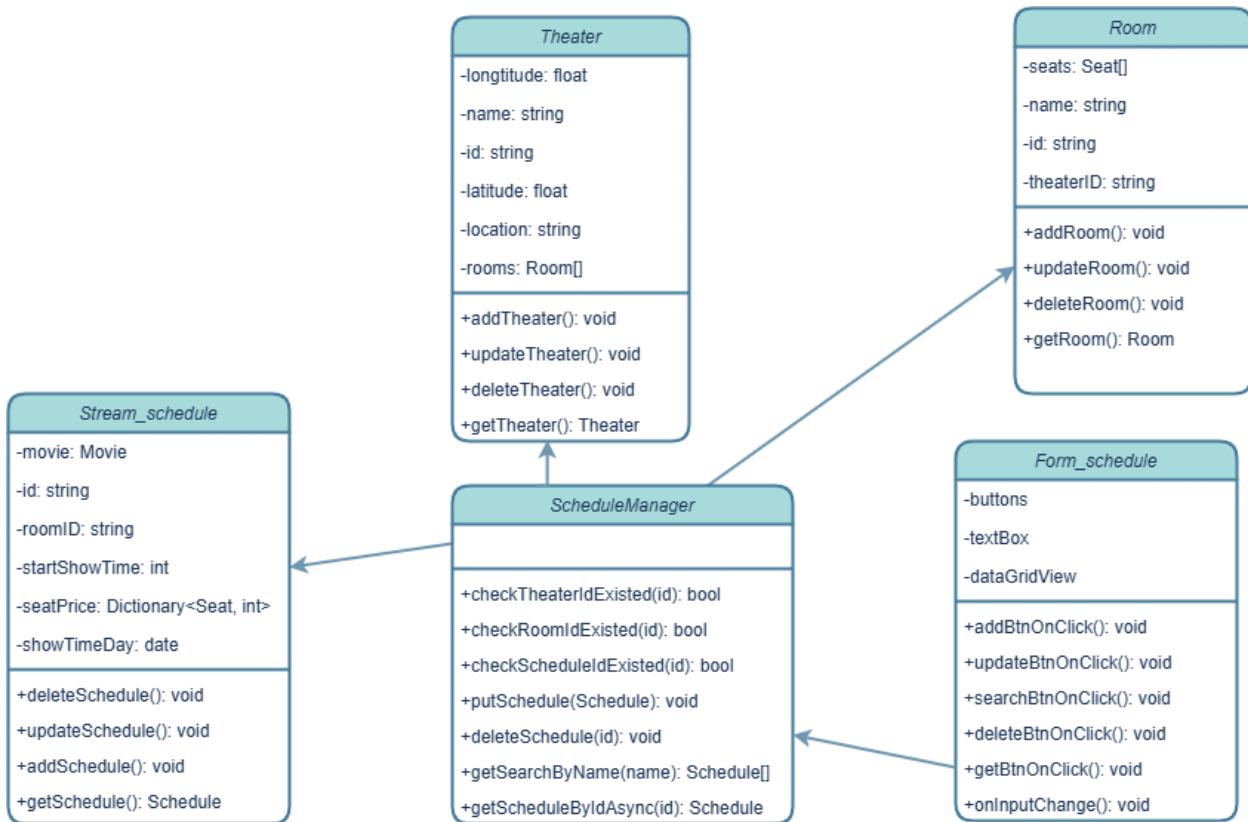


Figure 86 Thiết kế logic form quản lý lịch chiếu

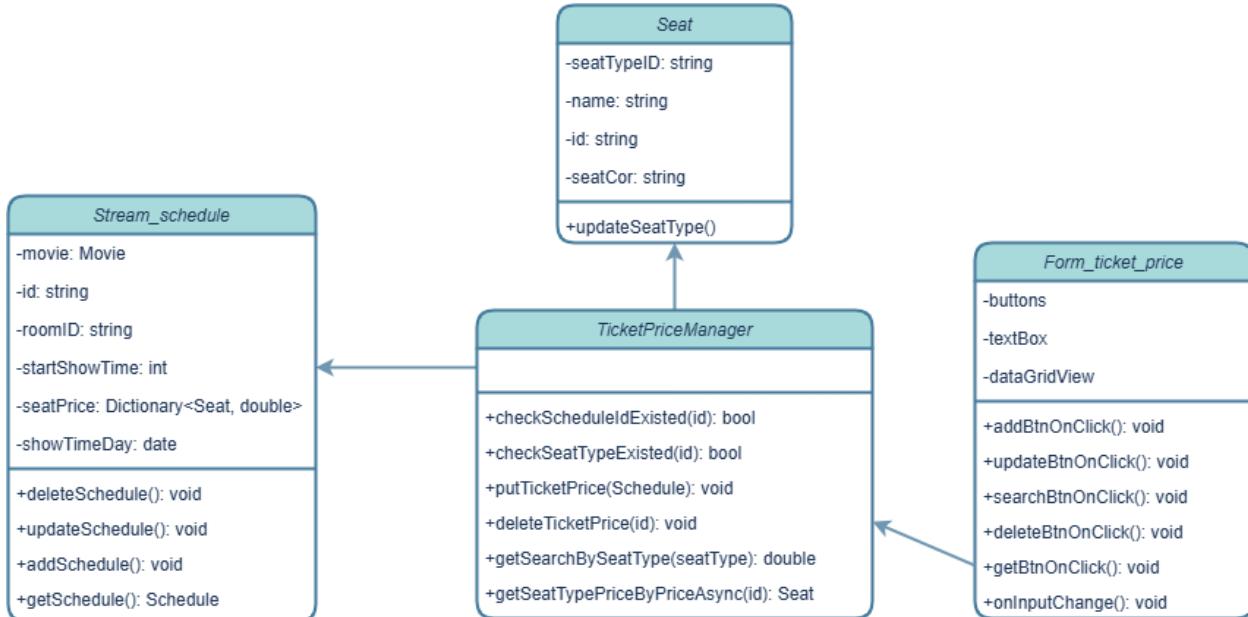


Figure 87 Thiết kế logic form quản lý giá vé

4.5 Thiết kế cơ sở dữ liệu

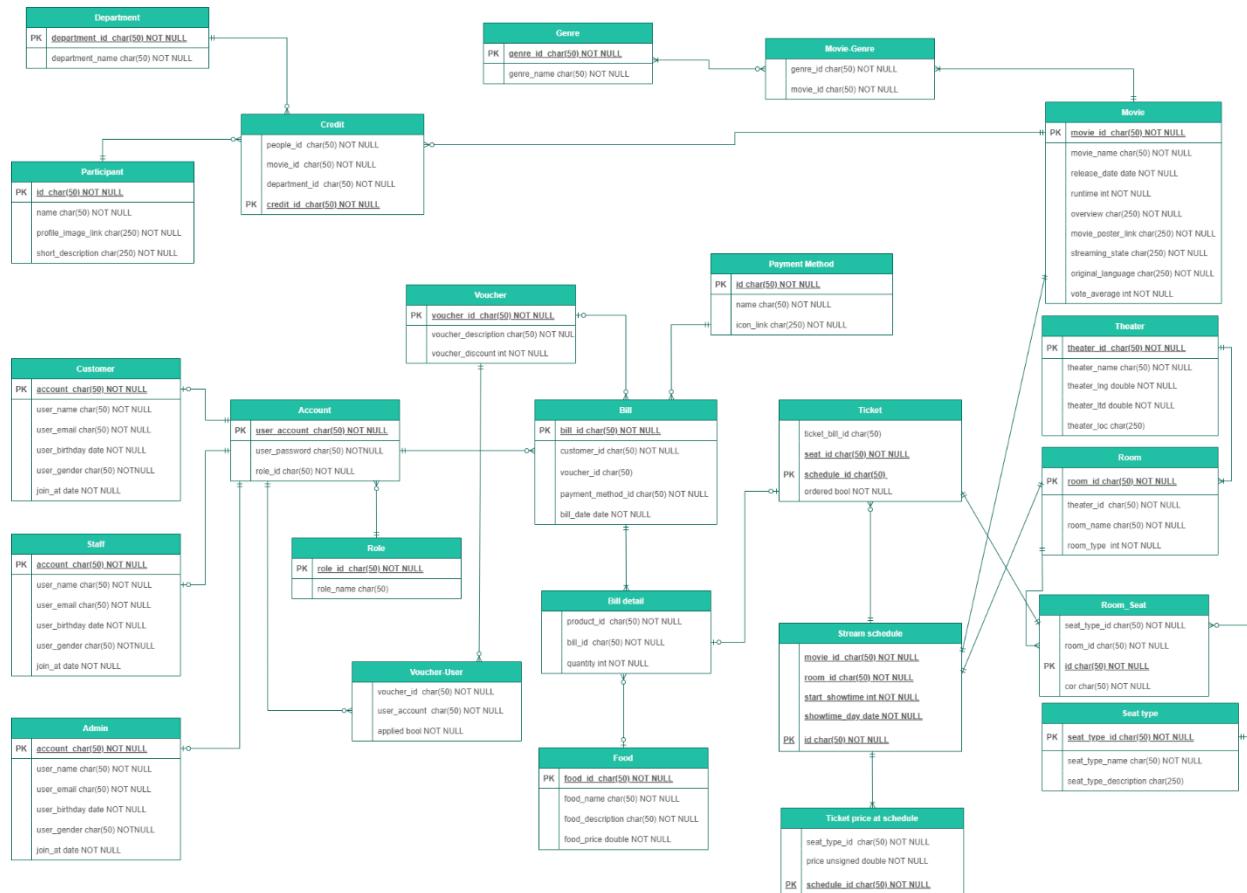


Figure 88 So đồ dữ liệu quan hệ

4.5.1 Chi tiết từng bảng

1. Bảng Account

Column name	Data type	Allow nulls
<u>user_account</u>	Nvarchar(50)	<input checked="" type="checkbox"/>
user_password	Nvarchar(50)	<input checked="" type="checkbox"/>
role_id	Nvarchar(50)	<input checked="" type="checkbox"/>

2. Bảng Role

Column name	Data type	Allow nulls
<u>role_id</u>	Nvarchar(50)	<input checked="" type="checkbox"/>
role_name	Nvarchar(50)	<input checked="" type="checkbox"/>

3. Bảng Customer

Column name	Data type	Allow nulls
<u>account</u>	Nvarchar(50)	<input checked="" type="checkbox"/>
user_name	Nvarchar(50)	<input checked="" type="checkbox"/>
user_email	Nvarchar(50)	<input checked="" type="checkbox"/>
user_birthday	datetime	<input checked="" type="checkbox"/>
user_gender	Nvarchar(50)	<input checked="" type="checkbox"/>
join_at	datetime	<input checked="" type="checkbox"/>

4. Bảng Staff

Column name	Data type	Allow nulls
<u>account</u>	Nvarchar(50)	<input checked="" type="checkbox"/>
user_name	Nvarchar(50)	<input checked="" type="checkbox"/>
user_email	Nvarchar(50)	<input checked="" type="checkbox"/>
user_birthday	datetime	<input checked="" type="checkbox"/>
user_gender	Nvarchar(50)	<input checked="" type="checkbox"/>
join_at	datetime	<input checked="" type="checkbox"/>

5. Bảng Admin

Column name	Data type	Allow nulls
<u>account</u>	Nvarchar(50)	<input checked="" type="checkbox"/>
user_name	Nvarchar(50)	<input checked="" type="checkbox"/>
user_email	Nvarchar(50)	<input checked="" type="checkbox"/>
user_birthday	datetime	<input checked="" type="checkbox"/>
user_gender	Nvarchar(50)	<input checked="" type="checkbox"/>
join_at	datetime	<input checked="" type="checkbox"/>

6. Bảng Movie

Column name	Data type	Allow nulls
<u>movie_id</u>	Nvarchar(50)	<input checked="" type="checkbox"/>
movie_name	Nvarchar(50)	<input checked="" type="checkbox"/>
release_date	datetime	<input checked="" type="checkbox"/>
runtime	int	<input checked="" type="checkbox"/>
overview	Nvarchar(250)	<input checked="" type="checkbox"/>
movie_poster_link	Nvarchar(250)	<input checked="" type="checkbox"/>
streaming_state	Nvarchar(250)	<input checked="" type="checkbox"/>
original_language	Nvarchar(250)	<input checked="" type="checkbox"/>
vote_average	double	<input checked="" type="checkbox"/>

7. Bảng Participant

Column name	Data type	Allow nulls

<u>id</u>	Nvarchar(50)	<input checked="" type="checkbox"/>
name	Nvarchar(50)	<input checked="" type="checkbox"/>
profile_image_link	Nvarchar(250)	<input checked="" type="checkbox"/>
short_description	Nvarchar(250)	<input checked="" type="checkbox"/>

8. Bảng Department

Column name	Data type	Allow nulls
<u>department_id</u>	Nvarchar(50)	<input checked="" type="checkbox"/>
department_name	Nvarchar(50)	<input checked="" type="checkbox"/>

9. Bảng Credit

Column name	Data type	Allow nulls
people_id	Nvarchar(50)	<input checked="" type="checkbox"/>
movie_id	Nvarchar(50)	<input checked="" type="checkbox"/>
department_id	Nvarchar(50)	<input checked="" type="checkbox"/>
<u>credit_id</u>	Nvarchar(50)	<input checked="" type="checkbox"/>

10. Bảng Genre

Column name	Data type	Allow nulls
<u>genre_id</u>	Nvarchar(50)	<input checked="" type="checkbox"/>
genre_name	Nvarchar(50)	<input checked="" type="checkbox"/>

11. Bảng Movie_Genre

Column name	Data type	Allow nulls
genre_id	Nvarchar(50)	<input checked="" type="checkbox"/>
movie_id	Nvarchar(50)	<input checked="" type="checkbox"/>

12. Bảng Theater

Column name	Data type	Allow nulls
<u>theater_id</u>	Nvarchar(50)	<input checked="" type="checkbox"/>
theater_name	Nvarchar(50)	<input checked="" type="checkbox"/>
theater_lng	Nvarchar(50)	<input checked="" type="checkbox"/>
theater_ltd	double	<input checked="" type="checkbox"/>
theater_loc	Nvarchar(250)	<input checked="" type="checkbox"/>

13. Bảng Room

Column name	Data type	Allow nulls
<u>room_id</u>	Nvarchar(50)	<input checked="" type="checkbox"/>
theater_id	Nvarchar(50)	<input checked="" type="checkbox"/>
room_name	Nvarchar(50)	<input checked="" type="checkbox"/>
room_type	int	<input checked="" type="checkbox"/>

14. Bảng Seat_type

Column name	Data type	Allow nulls
<u>seat_type_id</u>	Nvarchar(50)	<input checked="" type="checkbox"/>
seat_type_name	Nvarchar(50)	<input checked="" type="checkbox"/>
seat_type_description	Nvarchar(250)	<input checked="" type="checkbox"/>

15. Bảng Room_seat

Column name	Data type	Allow nulls
<u>id</u>	Nvarchar(50)	<input checked="" type="checkbox"/>
seat_type_id	Nvarchar(50)	<input checked="" type="checkbox"/>
room_id	Nvarchar(50)	<input checked="" type="checkbox"/>
cor	Nvarchar(50)	<input checked="" type="checkbox"/>

16. Bảng StreamSchedule

Column name	Data type	Allow nulls
<u>movie_id</u>	Nvarchar(50)	<input checked="" type="checkbox"/>
<u>room_id</u>	Nvarchar(50)	<input checked="" type="checkbox"/>
<u>start_showtime</u>	int	<input checked="" type="checkbox"/>
<u>showtime_day</u>	datetime	<input checked="" type="checkbox"/>
<u>id</u>	Nvarchar(50)	<input checked="" type="checkbox"/>

17. Bảng Ticket

Column name	Data type	Allow nulls
ticket_bill_id	Nvarchar(50)	<input checked="" type="checkbox"/>
<u>seat_id</u>	Nvarchar(50)	<input checked="" type="checkbox"/>
<u>schedule_id</u>	Nvarchar(50)	<input checked="" type="checkbox"/>
ordered	bool	<input checked="" type="checkbox"/>

18. Bảng Ticket_price_at_schedule

Column name	Data type	Allow nulls
seat_type_id	Nvarchar(50)	<input checked="" type="checkbox"/>
price	double	<input checked="" type="checkbox"/>
<u>schedule_id</u>	Nvarchar(50)	<input checked="" type="checkbox"/>

19. Bảng Food

Column name	Data type	Allow nulls
food_id	Nvarchar(50)	<input checked="" type="checkbox"/>
food_name	Nvarchar(50)	<input checked="" type="checkbox"/>
food_description	Nvarchar(50)	<input checked="" type="checkbox"/>
food_price	double	<input checked="" type="checkbox"/>

20. Bảng Bill

Column name	Data type	Allow nulls
bill_id	Nvarchar(50)	<input checked="" type="checkbox"/>
customer_id	Nvarchar(50)	<input checked="" type="checkbox"/>
voucher_id	Nvarchar(50)	<input checked="" type="checkbox"/>
payment_method_id	Nvarchar(50)	<input checked="" type="checkbox"/>
bill_date	datetime	<input checked="" type="checkbox"/>

21. Bảng BillDetail

Column name	Data type	Allow nulls
product_id	Nvarchar(50)	<input checked="" type="checkbox"/>
bill_id	Nvarchar(50)	<input checked="" type="checkbox"/>
quantity	int	<input checked="" type="checkbox"/>

22. Bảng Voucher

Column name	Data type	Allow nulls
voucher_id	Nvarchar(50)	<input checked="" type="checkbox"/>
voucher_description	Nvarchar(50)	<input checked="" type="checkbox"/>
voucher_discount	int	<input checked="" type="checkbox"/>

23. Bảng Voucher_user

Column name	Data type	Allow nulls
voucher_id	Nvarchar(50)	<input checked="" type="checkbox"/>
user_account	Nvarchar(50)	<input checked="" type="checkbox"/>
applied	bool	<input checked="" type="checkbox"/>

24. Bảng PaymentMethod

Column name	Data type	Allow nulls
id	Nvarchar(50)	<input checked="" type="checkbox"/>
name	Nvarchar(50)	<input checked="" type="checkbox"/>
icon_link	Nvarchar(250)	<input checked="" type="checkbox"/>

4.6 Phác họa giao diện người dùng

4.6.1 Giao diện khách hàng

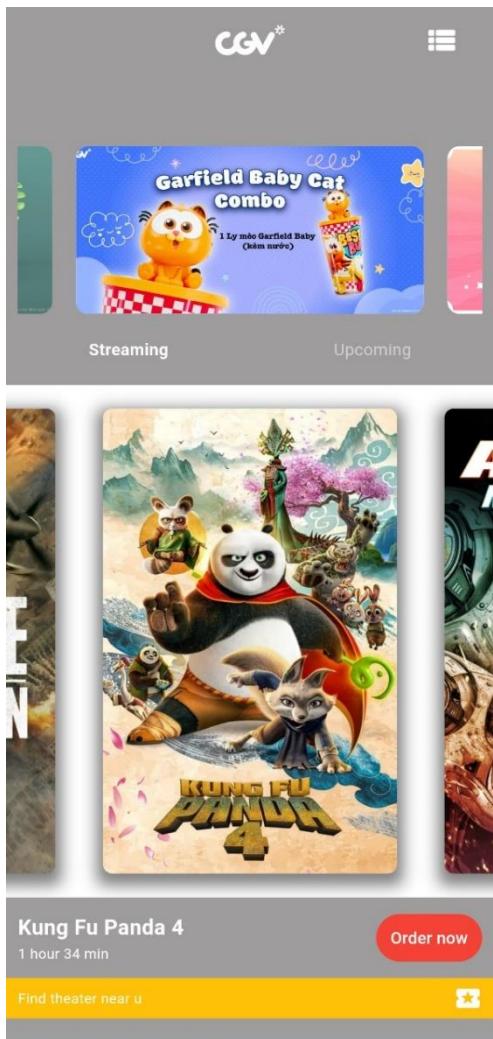


Figure 89 Giao diện trang chủ

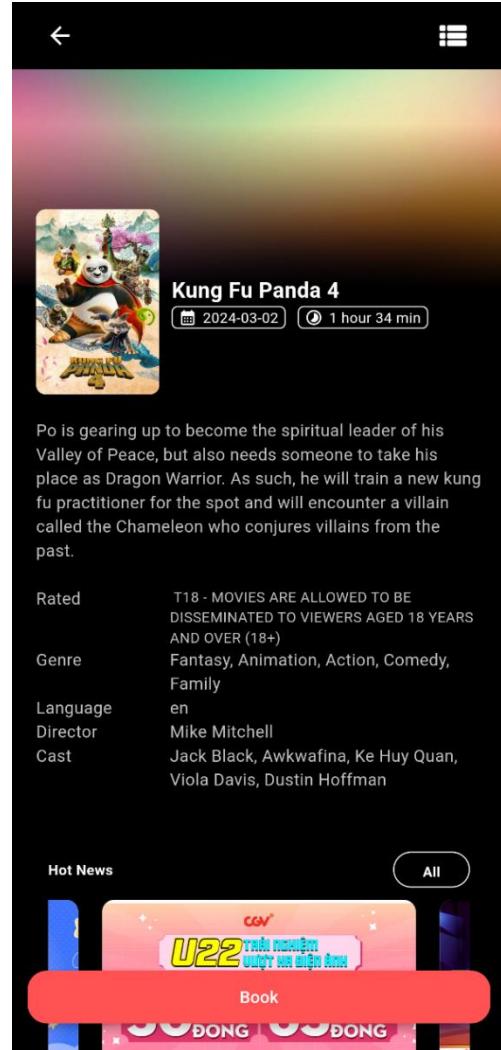


Figure 90 Trang thông tin chi tiết phim

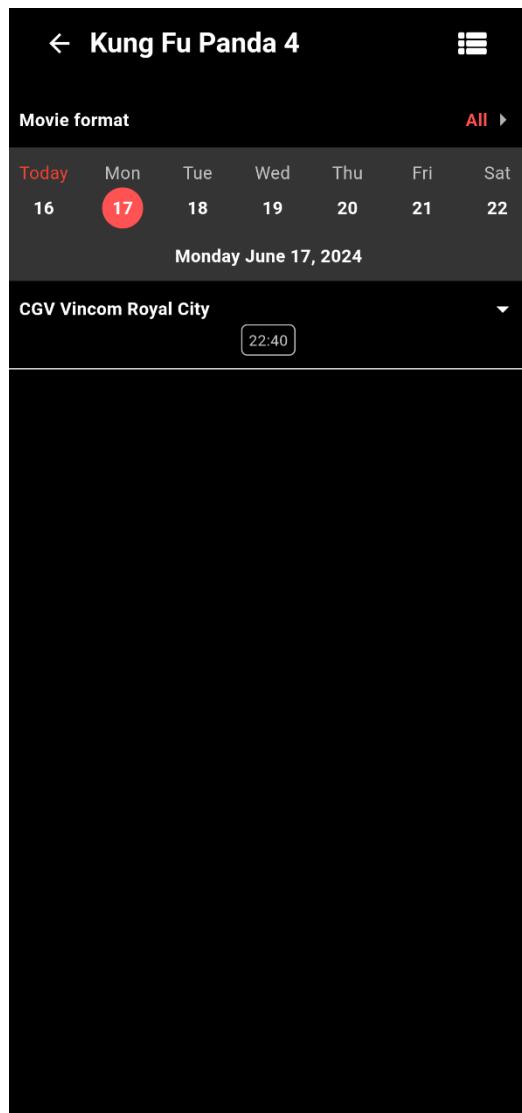


Figure 91 Trang chọn lịch chiếu

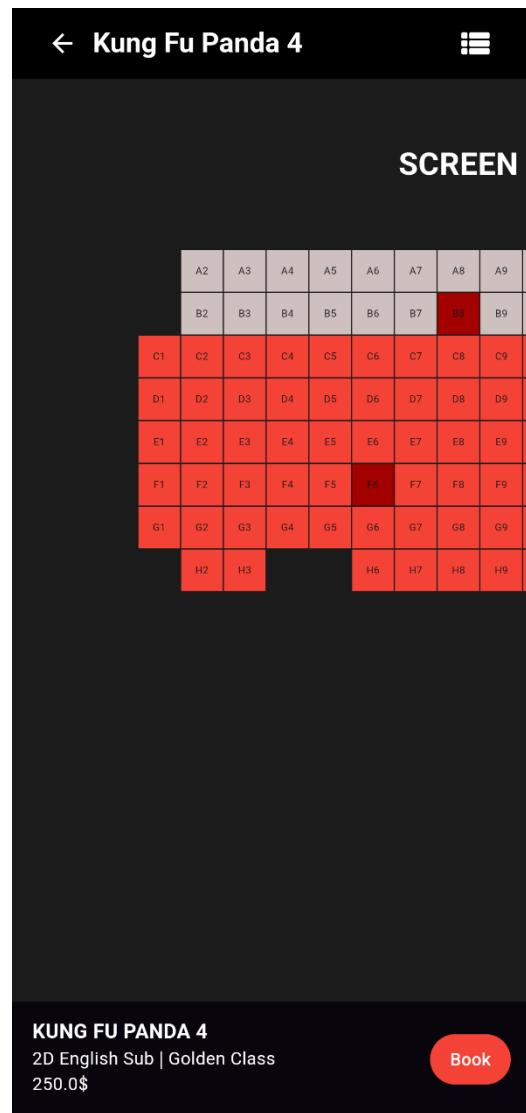


Figure 92 Trang chọn chỗ ngồi

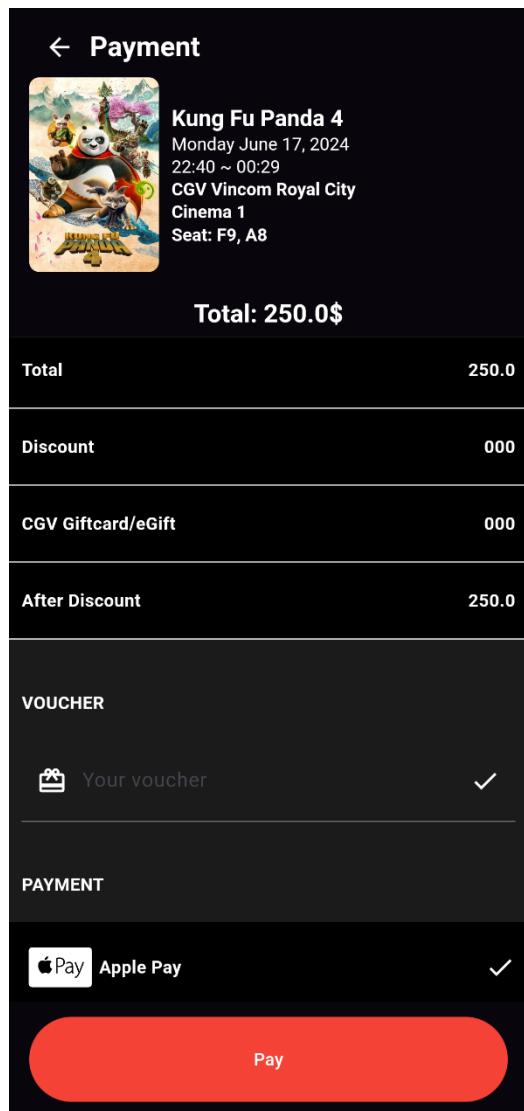


Figure 93 Trang thanh toán

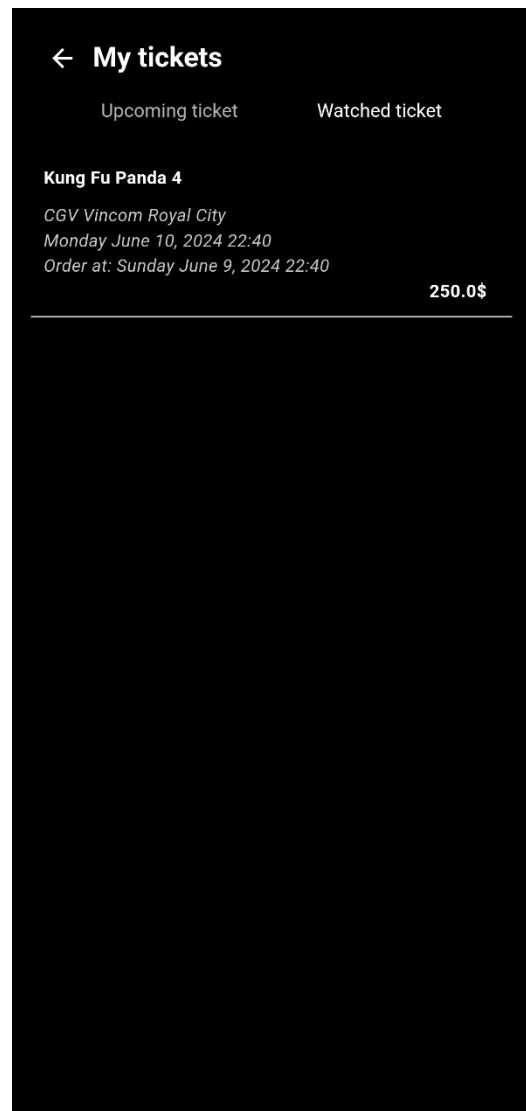


Figure 94 Trang xem hóa đơn

← Register

Email

Password

Fullname

Birthday

Gender

Male

Register

Figure 95 Trang đăng ký

← Log in

Email

Password

Log in

Register

Figure 96 Trang đăng nhập

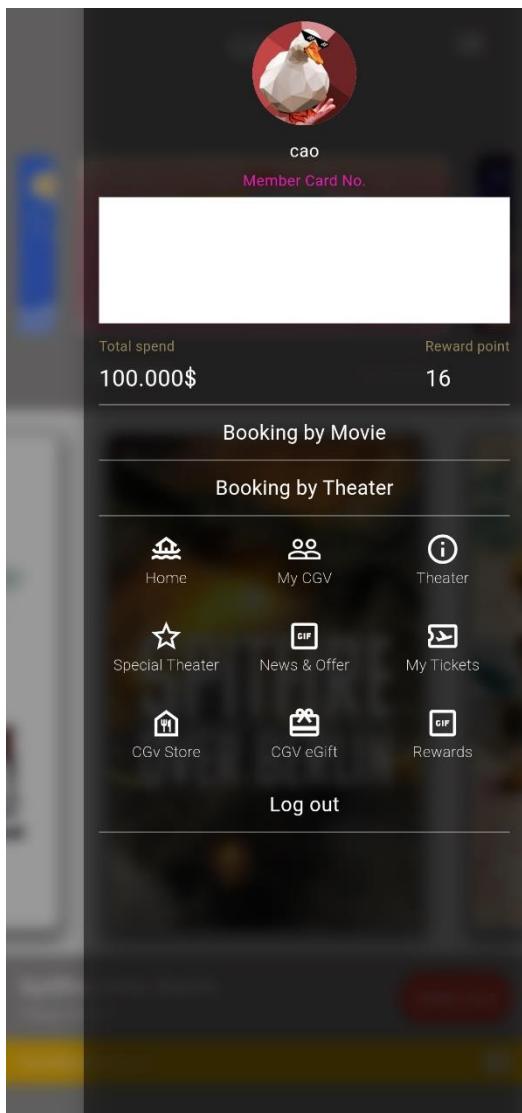


Figure 97 Menu

4.6.2 Giao diện quản trị

Theater Management x

Theater

ID	Name	Added_At	Loc	Lat	Lon
----	------	----------	-----	-----	-----

Search

ID Name
Latitude Longitude
Location

Figure 98 Trang quản lý rạp

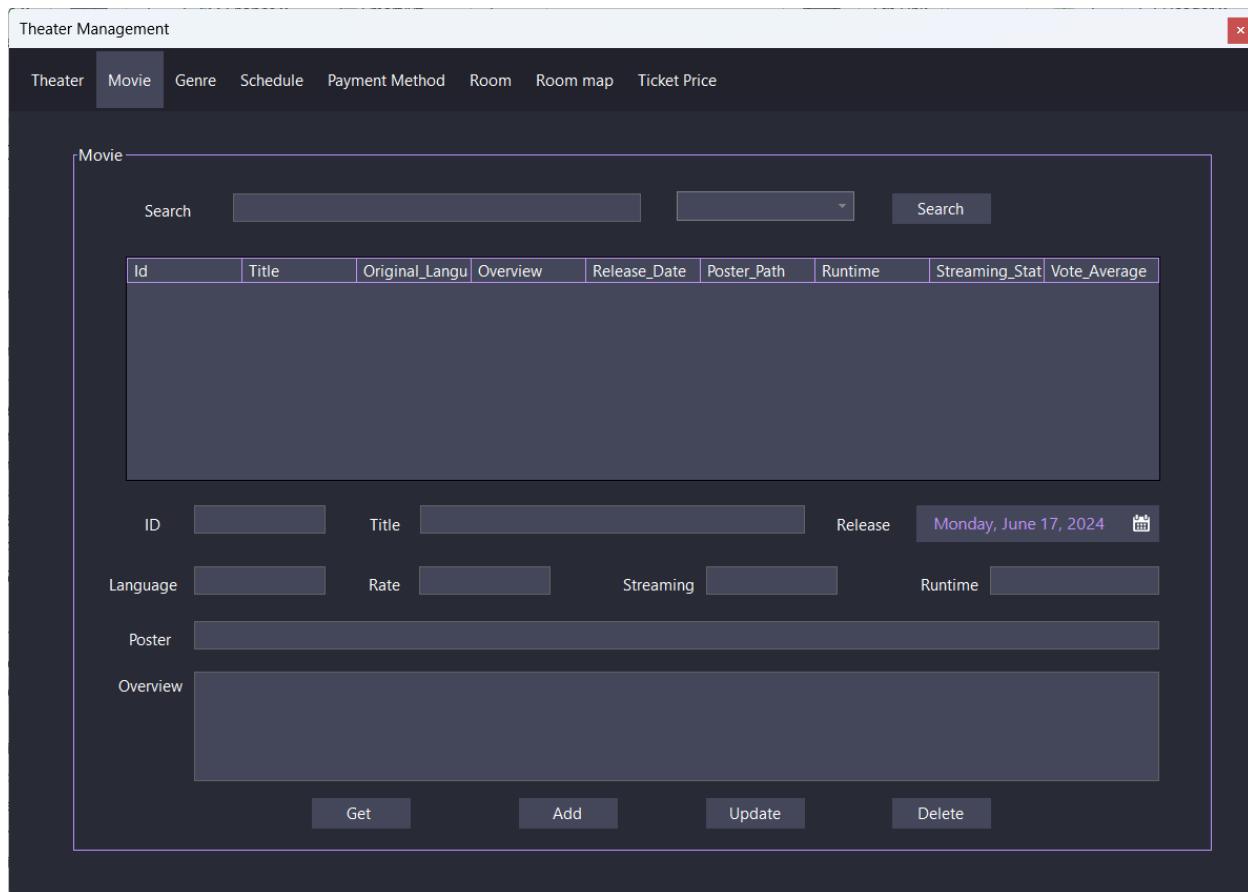


Figure 99 Trang quản lý phim

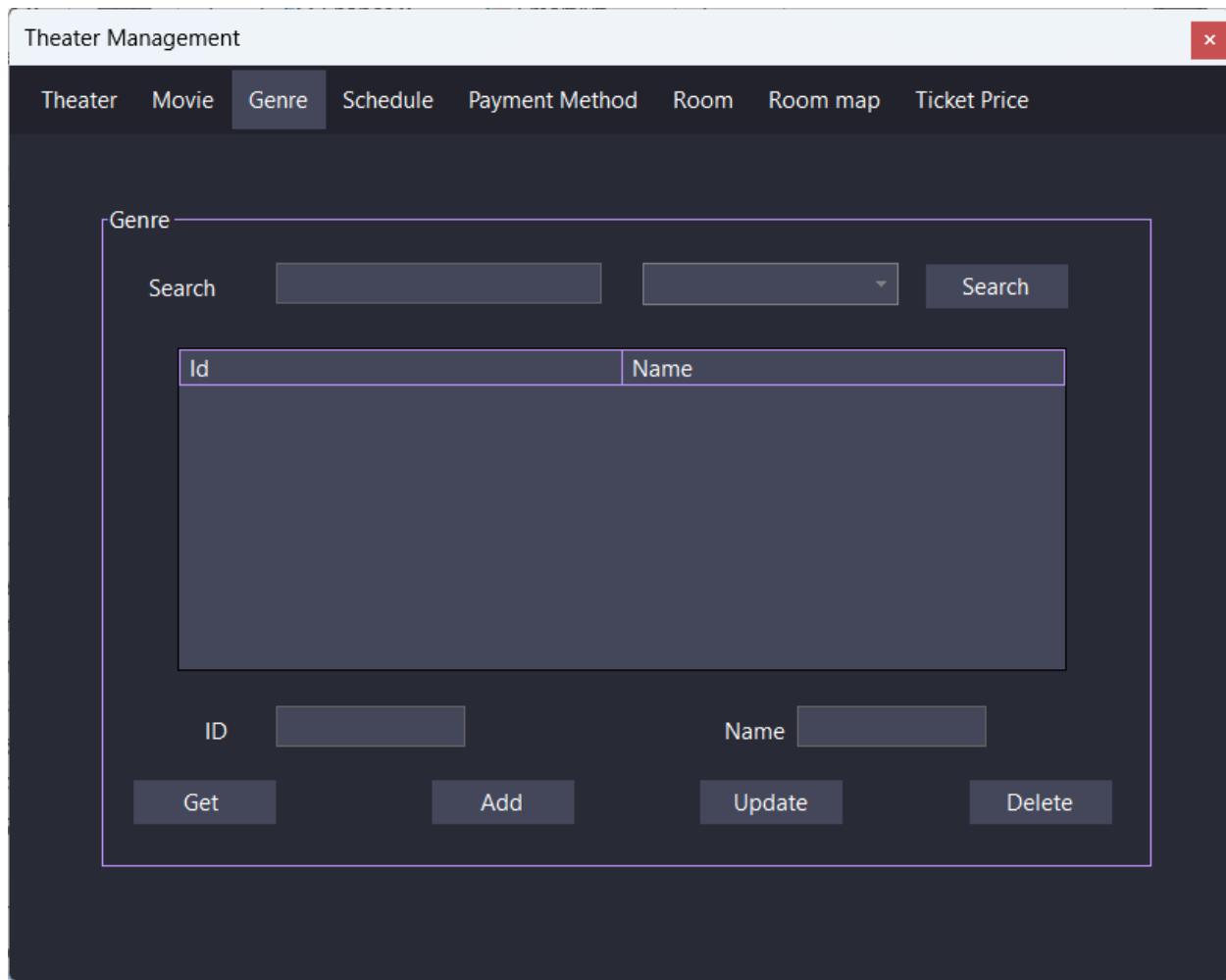


Figure 100 Trang quản lý thể loại

Theater Management

Schedule

ID	Movie ID	Room ID	Date	Time
----	----------	---------	------	------

Search

Movie ID

Room ID

Date 2024-06-17

Time 22:00

Get Add Update Delete

The screenshot shows a dark-themed user interface for managing theater schedules. At the top, there's a navigation bar with tabs: Theater, Movie, Genre, Schedule (which is highlighted in blue), Payment Method, Room, Room map, and Ticket Price. Below the navigation is a section titled "Schedule" containing a table with columns for ID, Movie ID, Room ID, Date, and Time. There are also dropdown menus for Movie ID and Room ID, and date/time pickers for Date and Time. At the bottom of this section are four buttons: Get, Add, Update, and Delete.

Figure 101 Trang quản lý lịch chiếu

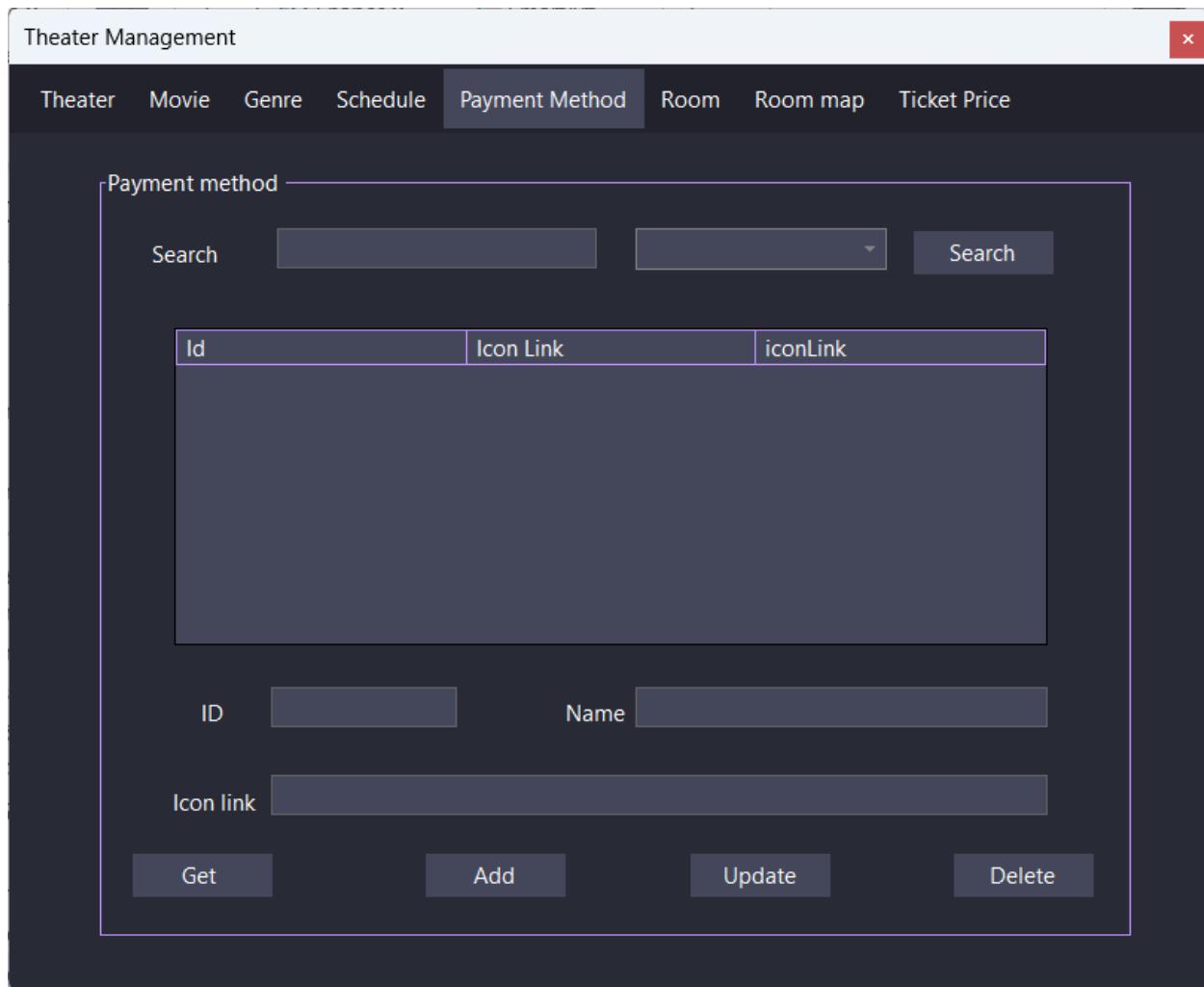


Figure 102 Trang quản lý phương thức thanh toán

Theater Management x

Theater Movie Genre Schedule Payment Method Room Room map Ticket Price

Room

Search

Id	Name	Theaterid	Room_Type

Room Index Theater ID

Room Type Name

Figure 103 Trang quản lý phòng chiếu

Theater Management x

Theater Movie Genre Schedule Payment Method Room Room map Ticket Price

Room map

Search

ID	Name	Roomid	Seattype

ID Room map

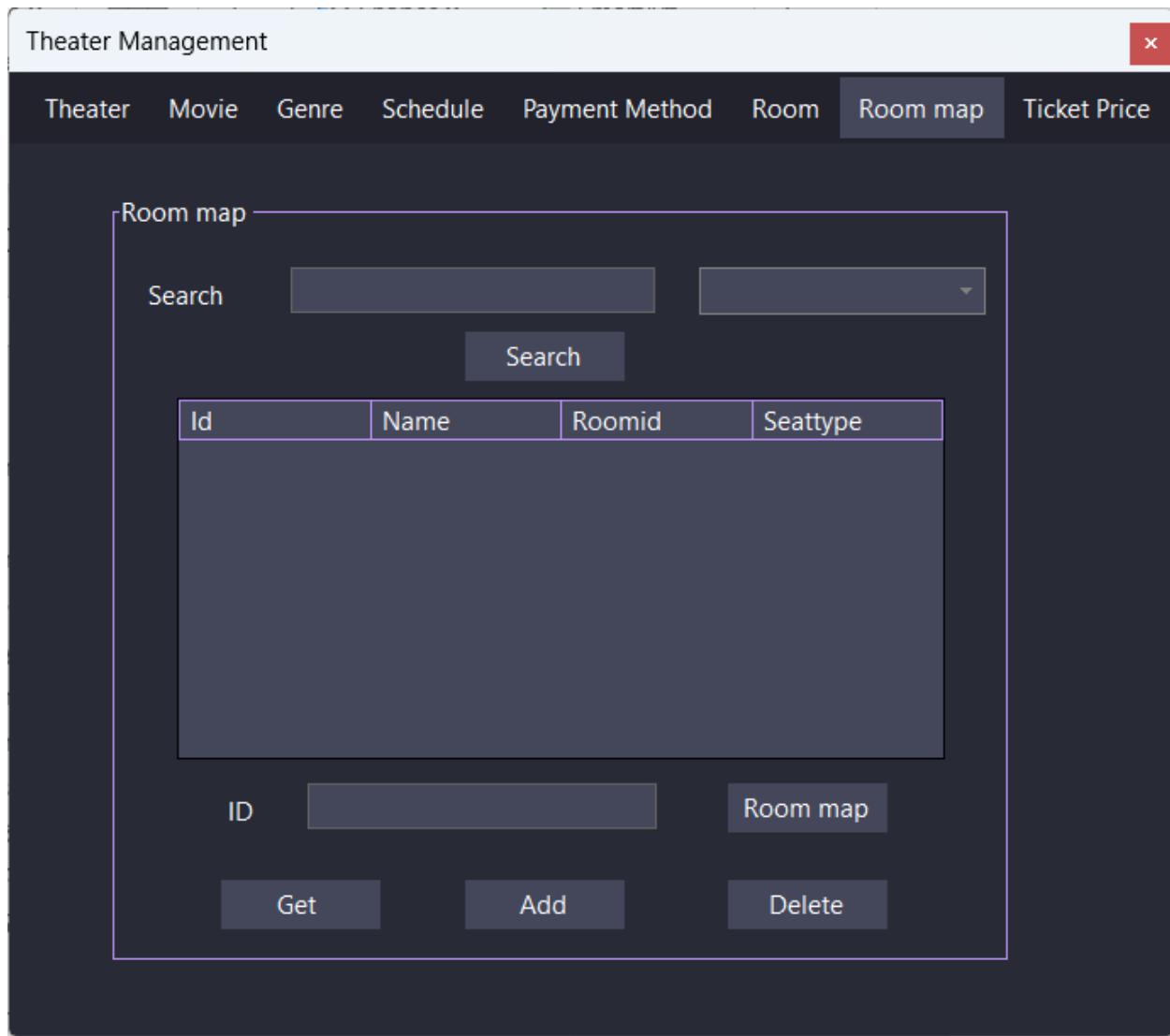


Figure 104 Trang quản lý sơ đồ phòng chiếu

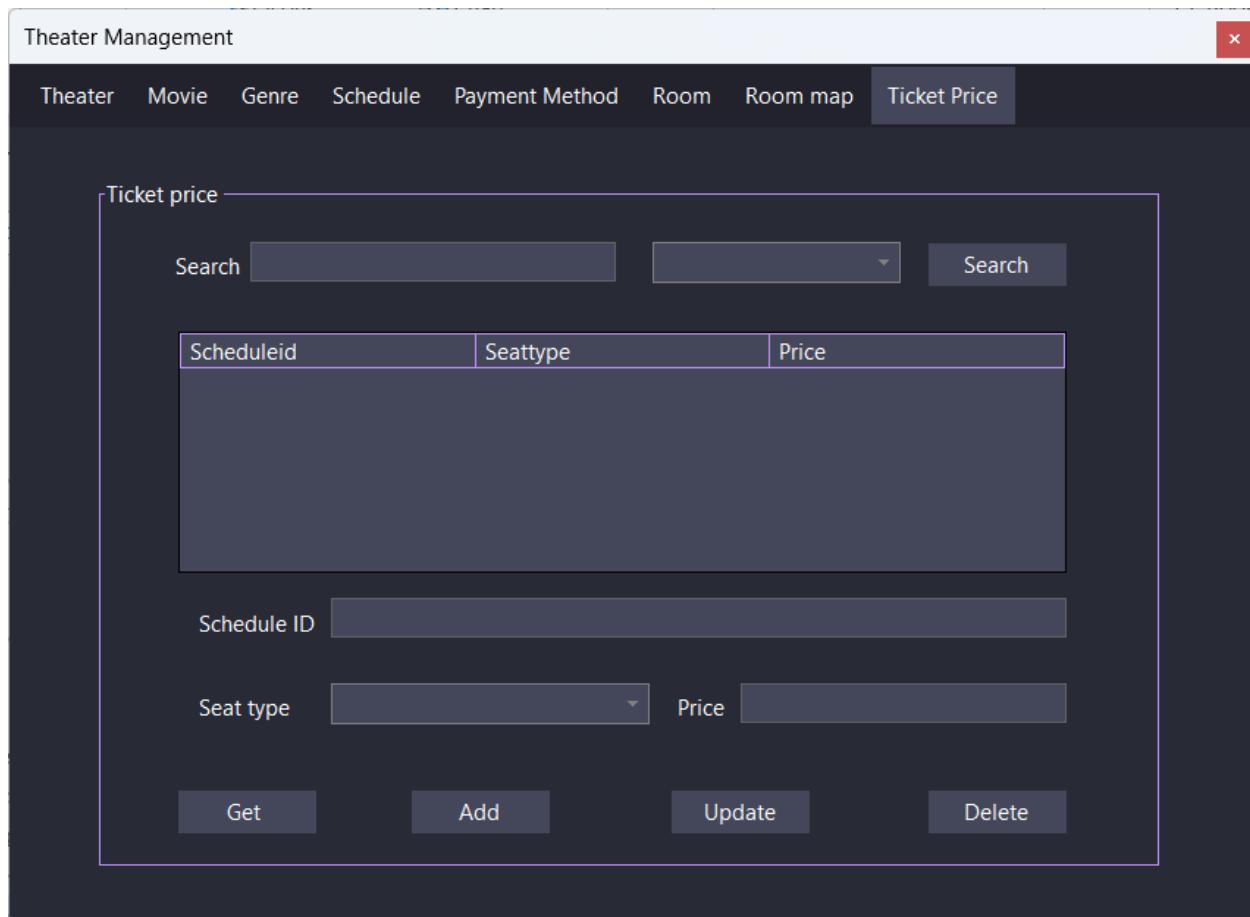


Figure 105 Trang quản lý giá vé

4.7 Áp dụng design pattern: Mẫu Strategy vào xử lí quy trình thanh toán

4.7.1 Mục tiêu của mẫu Strategy

Mẫu Strategy là một mẫu thiết kế thuộc nhóm hành vi cho phép xác định một nhóm thuật toán, đặt từng thuật toán vào một lớp riêng biệt và làm cho chúng có thể hoán đổi cho nhau. Strategy cho phép thuật toán biến đổi độc lập với các khách hàng sử dụng nó. Điều này giúp tăng cường tính mô-đun và tái sử dụng của mã, bởi vì nó tách rời việc triển khai của các thuật toán từ các lớp sử dụng chúng. [3] [4]

4.7.2 Vấn đề

Trong lập trình hướng đối tượng, các ứng dụng thường phải đối mặt với những thách thức liên quan đến việc lựa chọn hành vi thích hợp trong thời gian chạy. Một ví dụ điển hình là việc xử lý các chiến lược thanh toán khác nhau trong một hệ thống thương mại điện tử. Khi không sử dụng Strategy Pattern, việc thêm hoặc thay đổi các phương thức thanh toán có thể yêu

cần sửa đổi lớn trong mã nguồn, dẫn đến việc vi phạm nguyên tắc Mở - Đóng (Open-Closed Principle), làm tăng sự phức tạp và khó khăn trong việc bảo trì.

4.7.3 Giải pháp

Strategy Pattern cung cấp một giải pháp cho vấn đề trên bằng cách định nghĩa một tập hợp các thuật toán, mỗi thuật toán được đóng gói trong một lớp riêng biệt với một interface chung. Điều này cho phép thuật toán có thể thay đổi độc lập với các client sử dụng nó. Trong ví dụ về hệ thống thanh toán, các chiến lược thanh toán khác nhau như Credit Card, PayPal, hoặc Bitcoin có thể được thực hiện như các lớp riêng biệt, giúp việc thêm hoặc sửa đổi các phương thức thanh toán trở nên dễ dàng và linh hoạt hơn.

Việc sử dụng Strategy Pattern giúp tăng cường tính mô đun hóa và tái sử dụng của mã. Nó cũng giúp giảm sự phụ thuộc giữa các lớp và tăng tính linh hoạt cho ứng dụng. Ngoài ra, pattern cũng giúp ứng dụng tuân thủ nguyên tắc Open-Closed, giúp dễ dàng mở rộng mà không cần sửa đổi mã nguồn hiện có.

Mặc dù Strategy Pattern mang lại nhiều lợi ích, nhưng việc sử dụng nó cũng có thể dẫn đến một số sự thỏa hiệp. Ví dụ, nó có thể gây ra sự phức tạp ban đầu khi cần phải thiết kế và triển khai các interface và lớp cụ thể. Ngoài ra, nếu có quá nhiều chiến lược, việc quản lý chúng có thể trở nên khó khăn. [3]

4.7.4 Cấu trúc của mẫu Strategy

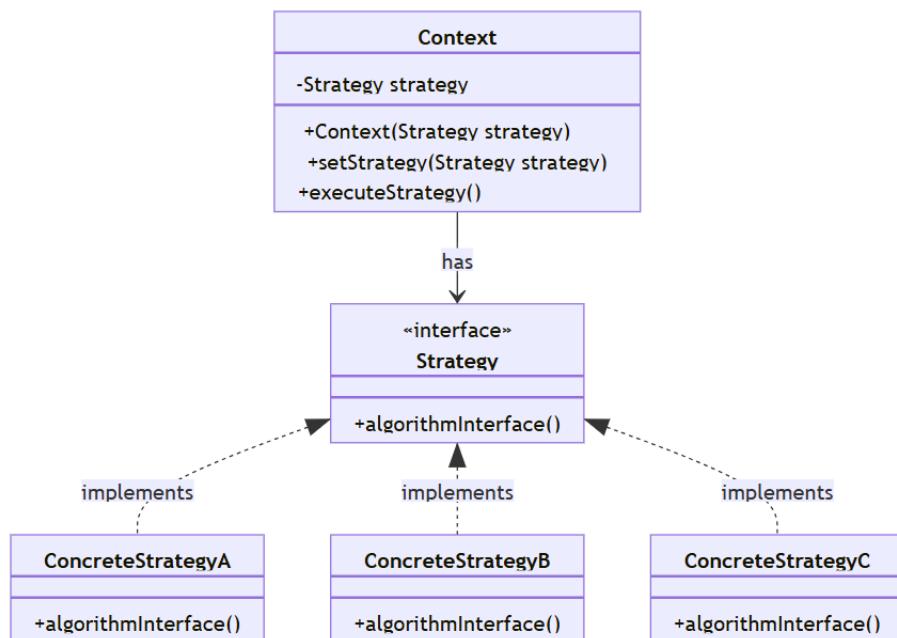


Figure 106 Cấu trúc của mẫu Strategy

- Context là lớp môi trường, nó chứa một tham chiếu đến một đối tượng Strategy. Context có thể thay đổi chiến lược tại thời điểm chạy.
- Strategy là một interface hoặc lớp trừu tượng định nghĩa một nhóm các thuật toán liên quan hoặc hành vi cụ thể. Trong mô hình Strategy, các thuật toán này đều được đóng gói và hoán đổi cho nhau một cách linh hoạt.
- ConcreteStrategyA, ConcreteStrategyB, và ConcreteStrategyC là các lớp triển khai cụ thể của interface Strategy. Mỗi một ConcreteStrategy triển khai một biến thể của một thuật toán hoặc một hành vi. [3]

4.7.5 Áp dụng vào xử lý quy trình thanh toán

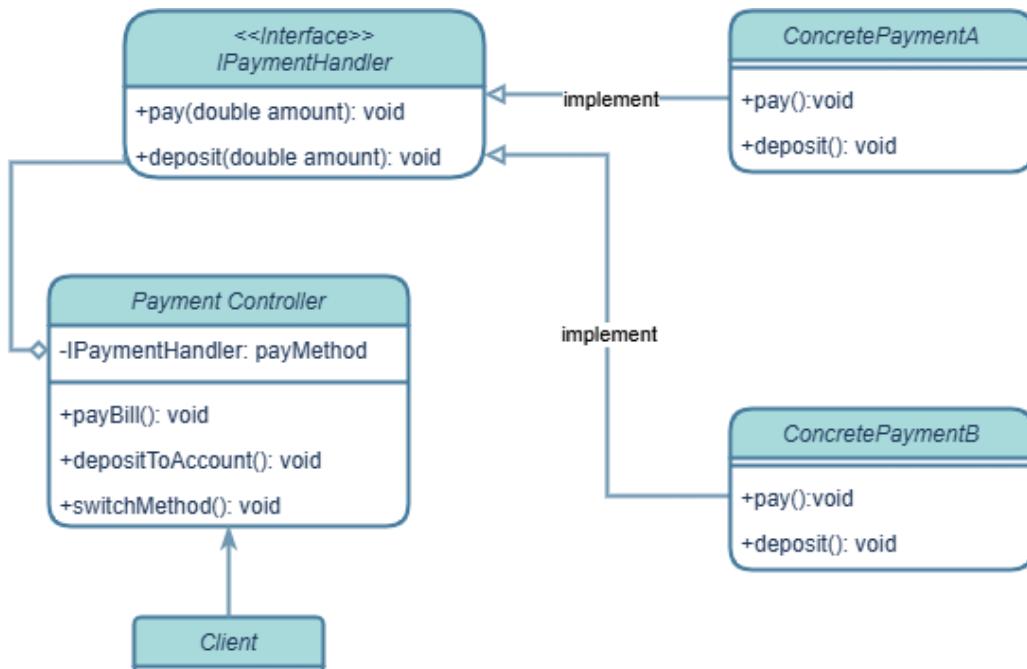


Figure 107 Áp dụng mẫu Strategy vào xử lý quy trình thanh toán

1 IPaymentHandler interface

Đây là interface cho các phương thức thanh toán khác nhau trong ứng dụng. Mỗi phương thức thanh toán sẽ cài đặt các hành động cụ thể.

2 ConcretePayment classes

Các lớp này sẽ hiện thực hóa interface trên và cài đặt cụ thể các hàm `pay()`, `deposit()`

3 PaymentController

Lớp này duy trì một tham chiếu đến một đối tượng `IPaymentMethod` và cho phép Client thay đổi phương thức thanh toán.

4 Client

Vậy lớp Client chỉ việc gọi `switchMethod()`, `payBill()` là có thể thanh toán, không cần quan tâm các lớp phương thức thanh toán cài đặt cụ thể ra sao, che giấu được độ phức tạp của thuật toán và đảm bảo tính đóng gói của OOP.

4.8 Chọn ngôn ngữ/framework lập trình

Nói chung, phát triển một ứng dụng di động là một nhiệm vụ phức tạp và đầy thách thức. Có nhiều framework có sẵn để phát triển ứng dụng di động. Android cung cấp một framework gốc dựa trên ngôn ngữ Java và iOS cung cấp một framework gốc dựa trên ngôn ngữ Objective-C / Swift.

Tuy nhiên, để phát triển một ứng dụng hỗ trợ cả hai hệ điều hành, chúng ta cần phải viết mã bằng hai ngôn ngữ khác nhau sử dụng hai framework khác nhau. Để giúp vượt qua sự phức tạp này, có các framework di động hỗ trợ cả hai hệ điều hành. Những framework này dao động từ framework ứng dụng di động hybrid đơn giản dựa trên HTML (sử dụng HTML cho giao diện người dùng và JavaScript cho logic ứng dụng) đến framework ngôn ngữ đặc thù phức tạp (thực hiện chuyển đổi mã sang mã gốc). Dù đơn giản hay phức tạp, các framework này luôn có nhiều nhược điểm, một trong những nhược điểm chính là hiệu suất chậm.

Trong tình huống này, Flutter - một framework đơn giản và hiệu suất cao dựa trên ngôn ngữ Dart, cung cấp hiệu suất cao bằng cách trực tiếp hiển thị giao diện người dùng trong canvas của hệ điều hành thay vì thông qua framework gốc. [5]

4.8.1 Giới thiệu

Flutter là SDK di động của Google để xây dựng ứng dụng gốc cho iOS và Android, Desktop (Windows, Linux, macOS), và Web từ một mã nguồn duy nhất. Khi xây dựng ứng dụng với Flutter, mọi thứ đều xoay quanh các Widget – các khối mà từ đó ứng dụng Flutter được xây dựng. Chúng là các phần tử cấu trúc đi kèm với nhiều chức năng cụ thể của material design và các widget mới có thể được tạo ra từ các widget hiện có. Quá trình kết hợp các widget với nhau được gọi là composition. Giao diện người dùng của ứng dụng được tạo thành từ nhiều widget đơn giản, mỗi widget đảm nhiệm một nhiệm vụ cụ thể. Đó là lý do tại sao các nhà phát triển Flutter thường nghĩ về ứng dụng Flutter của họ như một cây các widget.

là một cấu trúc mã nguồn mở để tạo ra các ứng dụng di động chất lượng cao, hiệu suất cao trên các hệ điều hành di động – Android và iOS. Nó cung cấp một SDK đơn giản, mạnh mẽ, hiệu quả và dễ hiểu để viết ứng dụng di động bằng ngôn ngữ riêng của Google, Dart.

4.8.2 Ưu điểm

Flutter đi kèm với các widget đẹp và có thể tùy chỉnh cho hiệu suất cao. Nó đáp ứng tất cả các nhu cầu và yêu cầu tùy chỉnh. Bên cạnh đó, Flutter còn mang lại nhiều ưu điểm khác như sau:

- Dart có một kho package lớn giúp việc lập trình dễ dàng hơn
- Các nhà phát triển chỉ cần viết một mã nguồn duy nhất cho cả hai ứng dụng (cả nền tảng Android và iOS). Flutter có thể được mở rộng cho các nền tảng khác trong tương lai.
- Flutter cần ít thử nghiệm hơn. Do có mã nguồn duy nhất, chỉ cần viết các bài kiểm tra tự động một lần cho cả hai nền tảng là đủ.

- Sự đơn giản của Flutter khiến nó trở thành ứng cử viên tốt cho phát triển nhanh. Khả năng tùy chỉnh và mở rộng của nó càng làm cho nó mạnh mẽ hơn.
- Với Flutter, các nhà phát triển có toàn quyền kiểm soát các widget và bộ cục của chúng.
- Flutter cung cấp các công cụ phát triển tuyệt vời, với khả năng hot reload ánh tượng.

4.8.3 Nhược điểm

Mặc dù có nhiều ưu điểm, Flutter vẫn có những nhược điểm sau:

- Vì được mã hóa bằng ngôn ngữ Dart, các nhà phát triển cần phải học một ngôn ngữ mới (mặc dù dễ học).
- Framework hiện đại có gắng tách biệt logic và giao diện người dùng càng tốt, nhưng trong Flutter, giao diện người dùng và logic lại được xen lấn. Chúng ta có thể khắc phục điều này bằng cách lập trình thông minh và sử dụng module cao cấp để tách biệt giao diện người dùng và logic.
- Flutter là một framework khác để tạo ứng dụng di động. Các nhà phát triển gặp khó khăn trong việc lựa chọn công cụ phát triển phù hợp trong một lĩnh vực đồng đúc.

Phần 5. Lập trình ứng dụng

5.1 Nền tảng cơ sở dữ liệu (Firebase)

Firebase là một sản phẩm của Google giúp các nhà phát triển xây dựng, quản lý và phát triển ứng dụng của họ một cách dễ dàng. Firebase giúp các nhà phát triển xây dựng ứng dụng của họ nhanh hơn và theo cách an toàn hơn. Firebase cung cấp dịch vụ cho android, ios, web và unity, cung cấp lưu trữ đám mây. Firebase là sử dụng kiến trúc NoSQL để lưu trữ cơ sở dữ liệu.

Ngoài ra, Firebase cũng hoàn toàn miễn phí để sử dụng các chức năng cơ bản như lưu trữ, truy vấn, xác thực... mà theo em là rất phù hợp với quy mô và phạm vi của chương trình em xây dựng. Nhược điểm duy nhất có lẽ là kiến trúc NoSQL mà em sẽ phải xoay sở một chút để áp dụng cơ sở dữ liệu SQL của mình vào. Tuy nhiên, với những tiện lợi mà nó đem lại, em quyết định sử dụng Firebase làm nơi lưu trữ dữ liệu trong chương trình này.

5.2 App quản trị (Windows)

Em đã sử dụng luôn các giao diện từ phần phác họa giao diện người dùng và thêm vào phần xử lí, vì vậy ở phần này em sẽ chỉ đưa ra vài giao diện chính của chương trình hoạt động và link github project.

Cấu trúc code được chia thành 3 phần: phần dữ liệu, phần giao diện và phần xử lý. Mỗi phần quản lý lớp thực thể sẽ được chia thành một form, các form này sẽ đi kèm với một class controller để thực hiện các thao tác phức tạp. Các thao tác này được gói lại trong một function và sẽ được gọi đến trong lớp form để tách rõ phần giao diện và phần xử lý logic, giúp giảm độ phức tạp của các class và dễ dàng bảo trì về sau. Các lớp dữ liệu sẽ chủ yếu được dùng để kéo dữ liệu từ cloud về và đẩy dữ liệu lên cloud qua các bước serialization.

Ứng dụng đã có các chức năng cơ bản để quản lý cơ sở dữ liệu đủ để có thể giúp cho phía mobile app hoạt động ổn định. Một số chức năng đã cài đặt:

- Đẩy dữ liệu lên Firebase Realtime database
- Quản lý các thực thể (phim, rạp, phòng chiếu, thể loại, phuong thức thanh toán...)
- Disable toàn bộ form khi đang thực hiện các thao tác đồng bộ (await/async)
- Xác thực dữ liệu (xác thực kiểu dữ liệu, kiểm tra trùng lặp...)

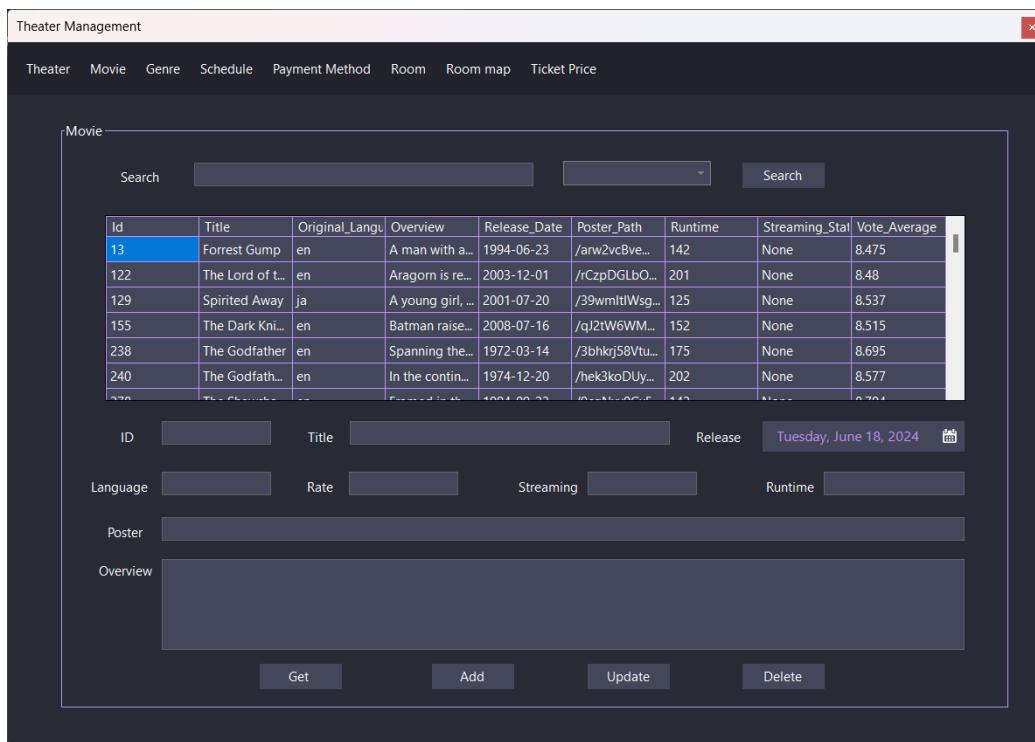


Figure 108 Thông kê phim có trong hệ thống

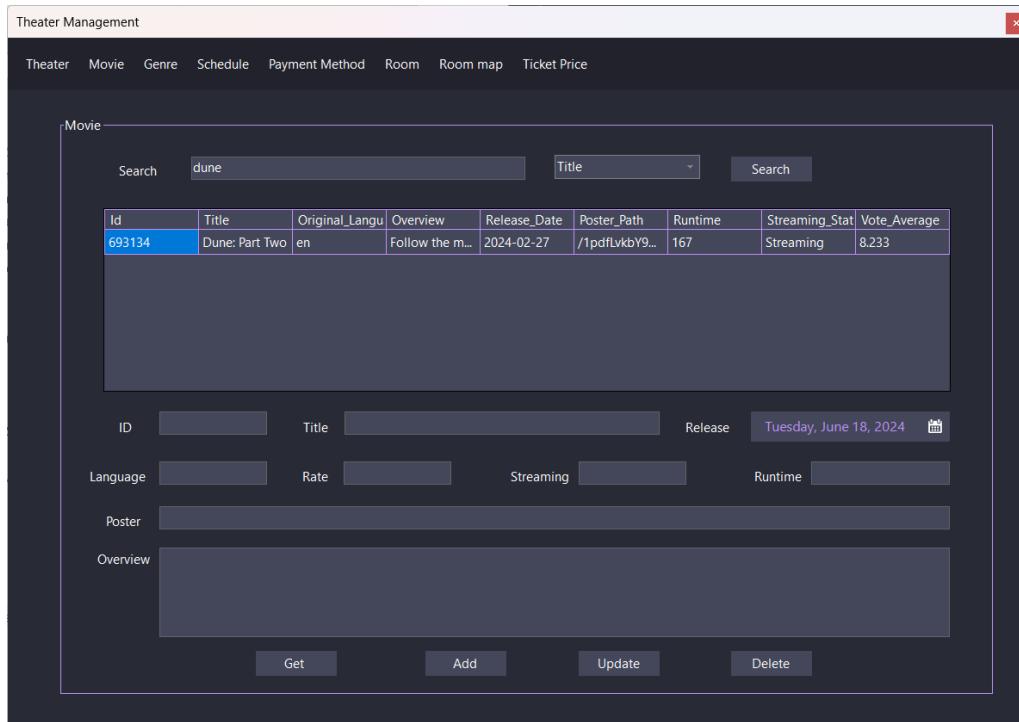


Figure 109 Chức năng tìm kiếm

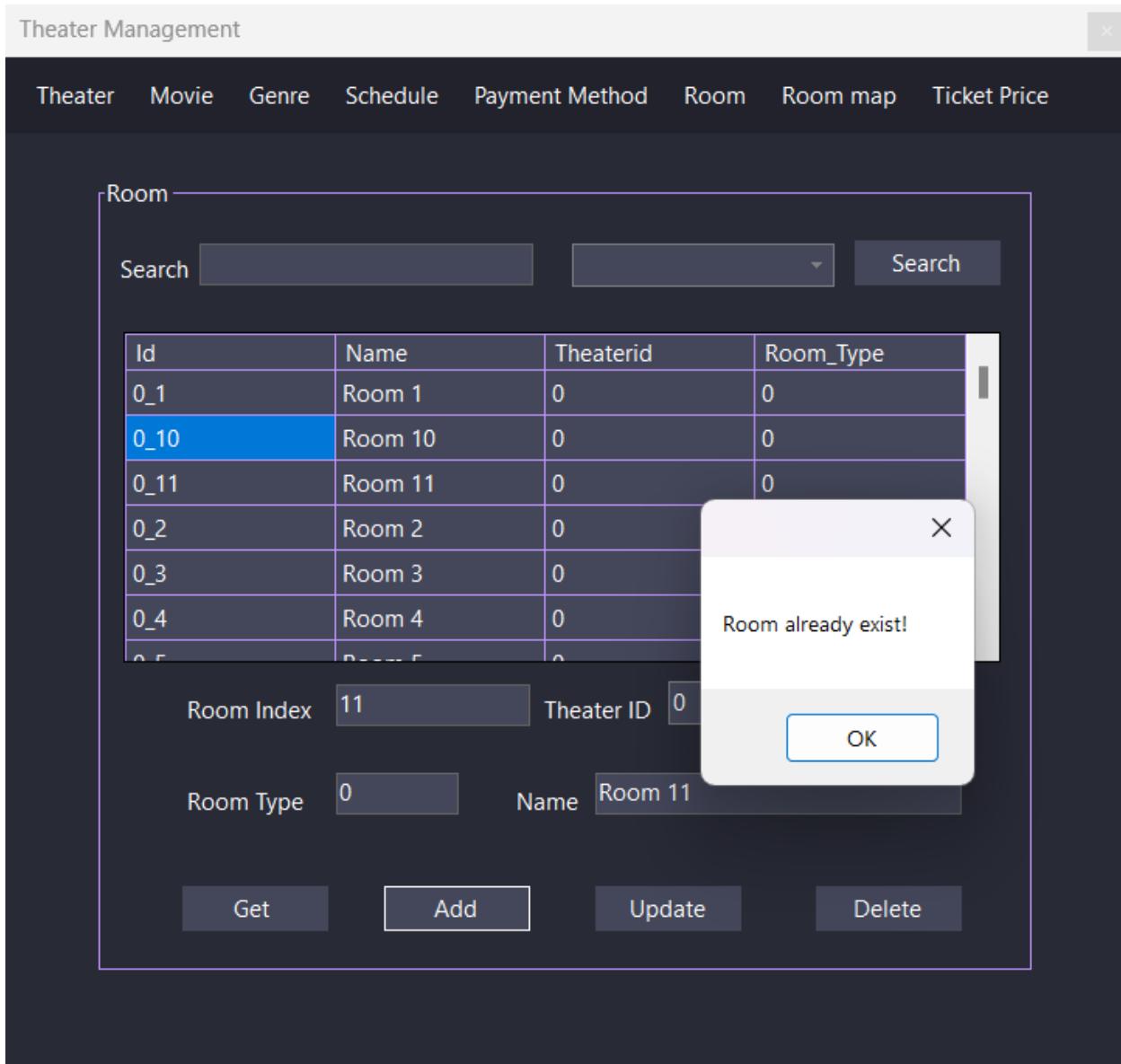


Figure 110 Kiểm tra dữ liệu đầu vào trước khi thêm vào hệ thống

Github project: [Movie Theater Management](#)

Ứng dụng đã đóng gói(.exe): [Winform App](#)

5.3 App client (Mobile)

Em đã sử dụng luôn các giao diện từ phần phác họa giao diện người dùng và thêm vào phần xử lý, vì vậy ở phần này em sẽ chỉ đưa ra vài giao diện chính của chương trình hoạt động và link github project.

Cấu trúc code được chia thành 3 phần: phần dữ liệu, phần giao diện và phần xử lý. Mỗi một trang sẽ được nhóm vào một folder, trong các folder sẽ chứa một folder widget con từ trang

chính phân rã ra, một class main_page chứa phần giao diện chính được lắp ráp từ các widget con và một class controller chịu trách nhiệm thực hiện các thao tác phức tạp gói thành một function và được gọi từ lớp giao diện. Các lớp dữ liệu sẽ chủ yếu được dùng để kéo dữ liệu từ cloud về và đẩy dữ liệu lên cloud qua các bước serialization.

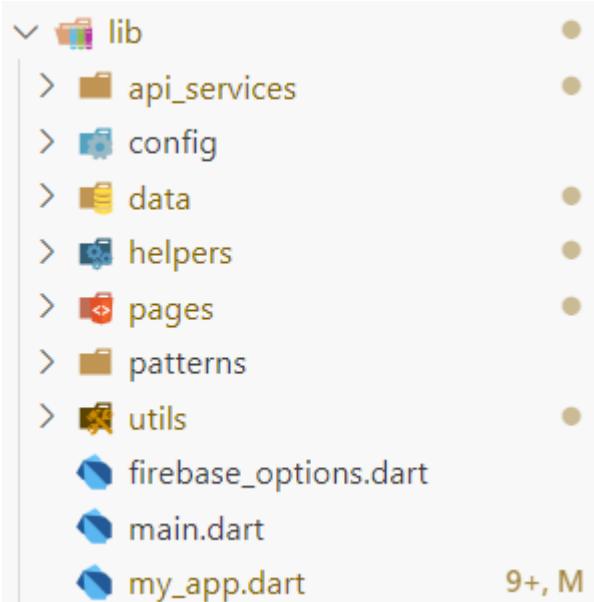


Figure 111 Cấu trúc project

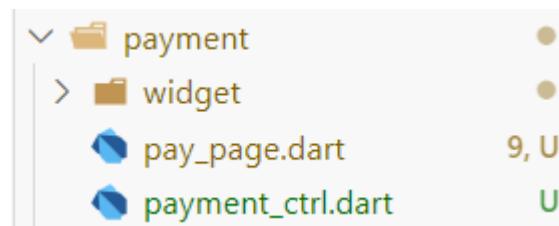


Figure 112 Cấu trúc code từng trang giao diện

Ứng dụng đã được cài đặt các giao diện cơ bản và cài đặt được chức năng chính và quan trọng nhất của ứng dụng đặt vé online là đặt vé. Ngoài ra, một số chức năng phụ cũng được cài đặt. Sau đây là tóm tắt những gì đã thực hiện được:

- Đẩy dữ liệu lên Firebase Realtime database khi người dùng đăng ký thông tin, mua vé và lấy dữ liệu về để hiển thị thông tin hiển thị cho người dùng.
- Hiển thị màn hình loading, đảm bảo người dùng không thực hiện các yêu cầu chồng chéo khi đang đồng bộ (async/await)
- Cài đặt được giao diện cơ bản là ồn (dễ nhìn, có bố cục rõ ràng, có thẻ trang trí thêm) và thực hiện các chức năng ồn định.
- Cài đặt được các chức năng tối thiểu của một ứng dụng đặt vé xem phim (đăng ký, đăng nhập, xem thông tin phim, đặt vé, xem hóa đơn...)

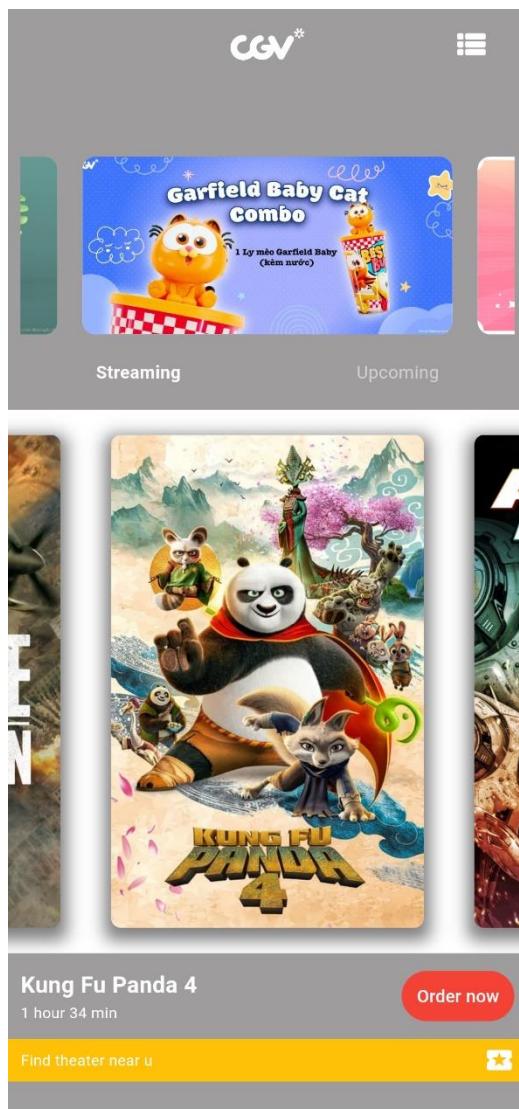


Figure 113 Trang chủ hiển thị danh sách phim, khuyến mãi

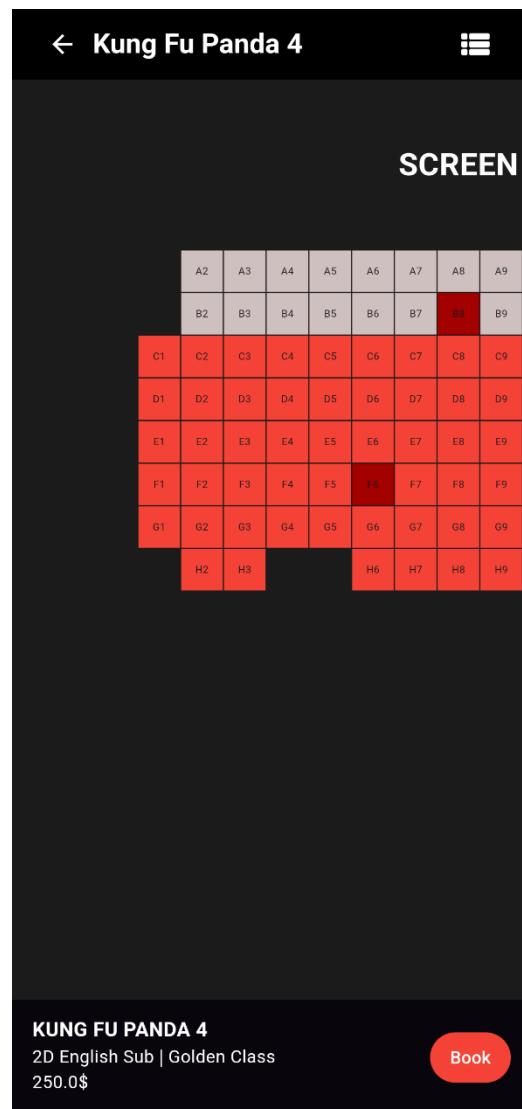


Figure 114 Giao diện chọn chỗ ngồi đặt vé

Đây là hai phần mà em cho là khó thực hiện nhất vì phải áp dụng một số phần phức tạp để cài đặt frontend và backend. Thiết kế ứng dụng được lấy cảm hứng từ ứng dụng CGV mobile.

Github Project: [Movie Theater Mobile App](#)

Ứng dụng đã đóng gói (.apk): [Movie Theater APK](#)

Phần 6. Kết luận

6.1 Kết quả đồ án

Tổng thể đồ án em đã đạt được một số kết quả sau:

1. Thực hiện cài đặt thành công cơ sở dữ liệu sử dụng Firebase Realtime database.
2. Thực hiện cài đặt được mobile app sử dụng framewrok Flutter, cài đặt một số chức năng quan trọng như xem chi tiết phim, đặt vé, đăng nhập, đăng ký...
3. Thực hiện cài đặt được một ứng dụng windows thực hiện quản lý cơ sở dữ liệu của rạp chiếu phim sử dụng C#
4. Thực hiện được một số biện pháp bảo mật cho ứng dụng như thực hiện băm mật khẩu trước khi gửi lên cloud, chặn đăng nhập từ địa chỉ IP đáng ngờ.
5. Ứng dụng được design pattern vào thiết kế hệ thống.
6. Trình bày rõ ràng các bước phân tích thiết kế hệ thống hướng đối tượng

6.2 Kỹ năng đạt được

Trong quá trình thực hiện đồ án, em đã học được một số kỹ năng hữu ích:

1. Kỹ năng học một ngôn ngữ lập trình mới.
2. Tìm hiểu các nội dung, tài liệu chuyên ngành từ nguồn tin cậy như tài liệu hướng dẫn của chính chủ, một số trang chia sẻ kiến thức uy tín.
3. Kỹ năng sử dụng các phương thức GET, POST, DELETE và sử dụng API để thu nhập dữ liệu.

6.3 Hướng phát triển

- Cài đặt phần frontend của ứng dụng cho đẹp mắt hơn.
- Cài đặt đầy đủ các chức năng cần thiết của ứng dụng đặt vé xem phim: cài đặt thêm chức năng áp dụng mã giảm giá, chức năng mua đồ ăn, chức năng thuê rạp theo nhóm...
- Triển khai cấu trúc dự án theo hướng module hơn để tăng tính mềm dẻo, tránh phụ thuộc giữa các lớp, dễ dàng bảo trì và nâng cấp.
- Tìm hiểu các cách lưu các key bảo mật hơn thay vì hiện tại là đang lưu trữ key trong các biến môi trường.
- Tìm hiểu và sử dụng nhiều framework để giảm độ phức tạp dự án.

Tài liệu tham khảo

- [1] T. Đ. Quế, Giáo trình phân tích và thiết kế hệ thống thông tin, Học viện công nghệ Bưu chính Viễn thông, 2017.
- [2] rishu_mishra, "Difference Between MVC and MVVM Architecture Pattern in Android," 16 5 2024. [Online]. Available: <https://www.geeksforgeeks.org/difference-between-mvc-and-mvvm-architecture-pattern-in-android/>. [Accessed 19 06 2024].
- [3] "Strategy | Design Pattern Tiếng Việt," 24 5 2024. [Online]. Available: <https://nguyenphuc22.github.io/Design-Patterns/strategy.html>. [Accessed 17 6 2024].
- [4] A. Shvets, "Strategy," [Online]. Available: <https://refactoring.guru/design-patterns/strategy>. [Accessed 17 6 2024].
- [5] "tutorialspoint," [Online]. Available: https://www.tutorialspoint.com/flutter/flutter_introduction.htm.
- [6] ANDY_NGO, "Tìm Hiểu Về Mô Hình Client - Server," 31 10 2020. [Online]. Available: <https://codelearn.io/sharing/tim-hieu-ve-mo-hinh-client-server>. [Accessed 17 6 2024].
- [7] "FlutterFire Overview | FlutterFire," [Online]. Available: <https://firebase.flutter.dev/docs/overview>. [Accessed 18 6 2024].
- [8] "firebase-database-dotnet," 20 9 2023. [Online]. Available: <https://github.com/step-up-labs/firebase-database-dotnet>. [Accessed 18 6 2024].