TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN TỐT NGHIỆP

Tên đề tài:

HỆ THỐNG RẠP CHIẾU PHIM TÍCH HỢP AI GỢI Ý

SVTH : Nguyễn Văn Tiến Dũng. 21TCLC-KHDL
GVHD: ThS. Mai Văn Hà

Đà Nẵng, 2025

TÓM TẮT

Tên đề tài: HỆ THỐNG RẠP CHIẾU PHIM TÍCH HỢP AI GỢI Ý
Sinh viên thực hiện: Nguyễn Văn Tiến Dũng
Số thẻ SV: 102210293 Lớp: 21TCLC-KHDL
Giảng viên hướng dẫn: Th.S Mai Văn Hà

Trong bối cảnh nhu cầu giải trí trực tuyến ngày càng gia tăng, việc xây dựng các hệ thống đặt vé xem phim thông minh và cá nhân hóa trở thành xu hướng tất yếu. Đề tài này tập trung nghiên cứu và phát triển một hệ thống đặt vé xem phim trực tuyến tích hợp trí tuệ nhân tạo (AI), cụ thể là mô hình học sâu LSTM (Long Short-Term Memory), nhằm nâng cao trải nghiệm người dùng thông qua việc tự động gợi ý các bộ phim phù hợp với sở thích cá nhân và hành vi lịch sử.

Hệ thống được xây dựng theo kiến trúc đa tầng, bao gồm: frontend sử dụng framework Nuxt.js (Vue 2) để xây dựng giao diện người dùng thân thiện; backend sử dụng Spring Boot để phát triển các API RESTful phục vụ cho quá trình xác thực, đặt vé và nhận gợi ý phim. Mô hình AI được huấn luyện bằng thư viện TensorFlow/Keras, kết hợp với dữ liệu lịch sử đặt vé, thể loại phim và đánh giá người dùng để đưa ra các đề xuất chính xác. Hệ thống còn tích hợp các công nghệ hiện đại như Redis để lưu mã xác nhận email, JWT để xác thực người dùng an toàn, và SMTP để gửi email xác nhận tài khoản.

Dữ liệu được lưu trữ và quản lý trên cơ sở dữ liệu MySQL, đảm bảo tính toàn vẹn và truy xuất hiệu quả. Hệ thống được triển khai toàn bộ trong môi trường container hóa thông qua Docker và Docker Compose, giúp đơn giản hóa việc triển khai và đảm bảo tính nhất quán trên các môi trường khác nhau. Mô hình LSTM sau khi huấn luyện đạt được chỉ số Cosine Similarity ~0.945, chứng tỏ khả năng gợi ý hiệu quả và chính xác.

Kết quả thực nghiệm cho thấy hệ thống vận hành ổn định, các chức năng cốt lõi được triển khai đầy đủ, và mô hình AI hoạt động hiệu quả trong việc cá nhân hóa nội dung gợi ý. Đề tài không chỉ đáp ứng các yêu cầu kỹ thuật đặt ra ban đầu mà còn mở ra nhiều hướng phát triển tiềm năng trong tương lai như tích hợp Transformer, xử lý dữ liệu thời gian thực, và triển khai trên nền tảng Kubernetes.

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: Nguyễn Văn Tiến Dũng Số thẻ sinh viên: 102210293

Lớp: 21TCLC-KHDL Khoa: Công Nghệ Thông Tin.

Ngành: Khoa Học Dữ Liệu & Trí Tuệ Nhân Tạo.

1. *Tên đề tài đồ án:*

HỆ THỐNG RẠP CHIẾU PHIM TÍCH HỢP AI GỢI Ý.

2. *Đề tài thuộc diện: ☐ Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*
3. *Các số liệu và dữ liệu ban đầu: TMDB movies dataset.*
4. *Nội dung các phần thuyết minh và tính toán:*
   - **Mở đầu:** Giới thiệu chung về đề tài, mục đích thực hiện, mục tiêu hướng đến, phạm vi, đối tượng tiếp cận, phương pháp nghiên cứu và cấu trúc đồ án.
   - **Chương 1**: Giới thiệu đề tài.Cơ sở lý thuyết. Giới thiệu tổng quan về các công nghệ và mô hình được sử dụng trong dự án.
   - **Chương 2**: Phân tích và thiết kế hệ thống. Phân tích các yêu cầu về chức năng và phi chức năng của hệ thống, sử dụng các sơ đồ để mô tả và thiết kế hệ thống.
   - **Chương 3**: Quy trình và công nghệ dùng để triển khai hệ thống, kết quả chức năng sau khi triển khai.
   - **Kết luận**: Tổng kết những kết quả đạt được cùng với hạn chế của hệ thống. Định hướng phát triển tương lai.

5. *Các bản vẽ, đồ thị (ghi rõ các loại và kích thước bản vẽ): Không có.*
6. *Họ tên người hướng dẫn:* Th.S Mai Văn Hà
7. *Ngày giao nhiệm vụ đồ án: ……/……/201…..*
8. *Ngày hoàn thành đồ án: ……/……/201…..*

Đà Nẵng, ngày tháng năm 201

Trưởng Bộ môn …….. **Người hướng dẫn**

LỜI NÓI ĐẦU

Công nghệ thông tin đang không ngừng phát triển và len lỏi vào mọi lĩnh vực của đời sống, trong đó có ngành giải trí và dịch vụ trực tuyến. Việc áp dụng trí tuệ nhân tạo vào các hệ thống phục vụ người dùng là hướng đi mang tính xu thế và thực tiễn. Với niềm đam mê về lĩnh vực lập trình ứng dụng và học máy, em đã lựa chọn đề tài "Hệ Thống Rạp Chiếu Phim Tích Hợp AI Gợi Ý" làm đề tài đồ án tốt nghiệp của mình, với mong muốn góp phần tìm hiểu và ứng dụng AI vào việc nâng cao trải nghiệm người dùng trong môi trường thực tế.

Trong suốt quá trình thực hiện đồ án, em đã nhận được sự hỗ trợ quý báu từ nhiều phía. Em xin chân thành cảm ơn quý thầy cô trong Khoa Công nghệ Thông tin, Trường Đại học Bách khoa – Đại học Đà Nẵng, những người đã trang bị cho em nền tảng kiến thức vững chắc và tạo điều kiện thuận lợi trong quá trình học tập và nghiên cứu.

Em đặc biệt tri ân Thầy Mai Văn Hà – người đã tận tình hướng dẫn, góp ý chuyên môn, và luôn theo sát tiến độ thực hiện đề tài. Những chỉ dẫn và định hướng từ Thầy là nguồn động lực to lớn giúp em hoàn thành đồ án này.

Mặc dù đã dành nhiều tâm huyết và cố gắng trong quá trình triển khai, nhưng do giới hạn về kinh nghiệm và thời gian, đồ án khó tránh khỏi những thiếu sót. Em rất mong nhận được những nhận xét và góp ý từ quý thầy cô để có thể hoàn thiện đề tài tốt hơn trong tương lai.

Em xin kính chúc quý thầy cô sức khỏe, thành công trong sự nghiệp giảng dạy và nghiên cứu.

CAM ĐOAN

Em xin cam đoan rằng toàn bộ nội dung trong đồ án tốt nghiệp này là kết quả làm việc nghiêm túc của riêng em, được thực hiện dưới sự hướng dẫn tận tình của thầy ThS. Mai Văn Hà. Trong suốt quá trình thực hiện, em đã tuân thủ đầy đủ các quy định về liêm chính học thuật và đạo đức nghiên cứu khoa học của Nhà trường.

Các tài liệu, số liệu và nội dung tham khảo sử dụng trong đồ án đều được em trích dẫn rõ ràng nguồn gốc, tên tác giả, thời gian và nơi công bố theo đúng quy cách trích dẫn tài liệu tham khảo.

Nếu phát hiện bất kỳ hành vi sao chép không hợp lệ hay vi phạm quy định học thuật nào trong nội dung đồ án, em xin hoàn toàn chịu trách nhiệm trước Hội đồng Khoa học và quy định của Trường Đại học Bách khoa – Đại học Đà Nẵng.

Sinh viên thực hiện

Nguyễn Văn Tiến Dũng

MỤC LỤC

DANH SÁCH CÁC BẢNG

DANH SÁCH CÁC HÌNH VẼ

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

Viết tắt	Tên đầy đủ	Diễn giải
AI	Artificial Intelligence	Trí tuệ nhân tạo – dùng để gợi ý phim dựa trên lịch sử đặt vé
LSTM	Long Short-Term Memory	Mạng nơ-ron học sâu để dự đoán chuỗi hành vi người dùng
API	Application Programming Interface	Giao diện lập trình ứng dụng – kết nối giữa frontend, backend và bên thứ ba
JWT	JSON Web Token	Chuẩn xác thực người dùng không trạng thái (stateless authentication)
SMTP	Simple Mail Transfer Protocol	Giao thức gửi email xác nhận tài khoản thông qua Gmail
TMDB	The Movie Database	Dịch vụ cung cấp dữ liệu phim miễn phí qua API
UI	User Interface	Giao diện người dùng
UX	User Experience	Trải nghiệm người dùng
SQL	Structured Query Language	Ngôn ngữ truy vấn dữ liệu trong cơ sở dữ liệu quan hệ (MySQL)
DB	Database	Cơ sở dữ liệu
RDBMS	Relational Database Management System	Hệ quản trị cơ sở dữ liệu quan hệ (MySQL)
CI/CD	Continuous Integration / Continuous Deployment	Tích hợp liên tục và triển khai liên tục trong quy trình DevOps
VPS	Virtual Private Server	Máy chủ ảo – nơi triển khai hệ thống thực tế
CRUD	Create, Read, Update, Delete	Các thao tác cơ bản với dữ liệu
RESTful API	Representational State Transfer API	Kiến trúc API được dùng trong hệ thống backend Spring Boot

IDE	Integrated Development Environment	Môi trường lập trình tích hợp (VSCode, IntelliJ IDEA...)
MVC	Model - View - Controller	Kiến trúc phần mềm được áp dụng trong backend Spring Boot
ORM	Object Relational Mapping	Kỹ thuật ánh xạ giữa đối tượng Java và bảng cơ sở dữ liệu (JPA, Hibernate)
SSR	Server-Side Rendering	Cơ chế kết xuất trang web phía server – được dùng trong Nuxt.js
CDN	Content Delivery Network	Mạng phân phối nội dung (có thể dùng cho ảnh phim lấy từ TMDB)
Colab	Google Colaboratory	Nền tảng máy tính đám mây để train mô hình AI (dùng LSTM)
JSON	JavaScript Object Notation	Định dạng trao đổi dữ liệu nhẹ giữa client và server
HTTPS	Hypertext Transfer Protocol Secure	Giao thức truyền dữ liệu có mã hóa
SSL/TLS	Secure Sockets Layer / Transport Layer Security	Giao thức bảo mật giúp truyền dữ liệu an toàn (kèm chứng chỉ khi dùng domain)
GPU	Graphics Processing Unit	Bộ xử lý đồ họa – tăng tốc huấn luyện mô hình AI

MỞ ĐẦU

1. Mục đích thực hiện:

Trong xu thế chuyển đổi số, nhu cầu số hóa các dịch vụ giải trí như đặt vé xem phim ngày càng trở nên phổ biến. Bên cạnh việc cung cấp trải nghiệm tiện lợi và nhanh chóng cho người dùng khi đặt vé, hệ thống hiện đại cần tích hợp trí tuệ nhân tạo để cá nhân hóa dịch vụ và tối ưu hóa trải nghiệm người dùng. Việc ứng dụng các thuật toán học sâu như LSTM vào phân tích hành vi người dùng, gợi ý nội dung phù hợp là một hướng tiếp cận hiệu quả.

Đề tài "Hệ Thống Rạp Chiếu Phim Tích Hợp AI Gợi Ý" được thực hiện nhằm mục đích xây dựng một hệ thống đặt vé xem phim trực tuyến với đầy đủ chức năng từ giao diện người dùng đến xử lý backend và đặc biệt là khả năng gợi ý phim thông minh dựa trên hành vi lịch sử. Hệ thống này vừa hỗ trợ tốt cho người dùng trong việc tìm kiếm phim yêu thích, vừa tạo nền tảng kỹ thuật để triển khai trong môi trường thực tế.

2. Mục tiêu đề tài:

Mục tiêu tổng quát:

Xây dựng thành công một hệ thống đặt vé xem phim trực tuyến tích hợp mô hình AI gợi ý phim, đảm bảo hoạt động ổn định, dễ sử dụng và có thể triển khai thực tế.

Mục tiêu cụ thể:

- Phát triển giao diện đặt vé bằng Nuxt.js (Vue 2) với trải nghiệm người dùng thân thiện.
- Phát triển trang admin quản lý hành động đặt vé, hủy vé của người dùng.
- Xây dựng backend API RESTful với Spring Boot, tích hợp xác thực bảo mật bằng JWT.
- Huấn luyện và triển khai mô hình AI LSTM để đưa ra các gợi ý phim chính xác dựa trên lịch sử đặt vé của người dùng
- Sử dụng MySQL để lưu trữ dữ liệu chính, Redis để lưu mã xác thực email.

Đảm bảo hệ thống hoạt động ổn định trong môi trường container hóa bằng Docker và hỗ trợ gửi email xác thực qua SMTP**.**

3. Phạm vi nghiên cứu:

Đề tài tập trung vào việc phát triển một hệ thống đặt vé xem phim trực tuyến hoạt động trên nền web, tích hợp đầy đủ quy trình từ đăng ký, xác thực email, đăng nhập, chọn phim, đặt ghế, quản lý lịch sử đến gợi ý phim bằng AI. Trong khuôn khổ đồ án, phần quản trị viên được thiết kế ở mức cơ bản và chưa triển khai đầy đủ.

4. Đối tượng nghiên cứu:

Hệ thống phục vụ hai nhóm đối tượng chính:

- Người dùng (Customer): có thể đăng ký tài khoản, xác thực qua email, đăng nhập, xem phim đang chiếu, chọn suất chiếu, đặt ghế, và nhận gợi ý phim.
- Quản trị viên(Admin): Có thể theo giõi vé, phòng, ghế và xác nhận các hành động đặt vé, hủy vé của người dùng.
- Mô hình AI: phân tích lịch sử đặt vé, thể loại phim và hành vi người dùng để đưa ra gợi ý phù hợp.

5. Phương pháp nghiên cứu:

- **Phân tích và khảo sát:** tìm hiểu các hệ thống đặt vé hiện có, xác định nhu cầu và hành vi người dùng.
- **Thiết kế hệ thống:** xây dựng kiến trúc tổng thể bao gồm frontend, backend, cơ sở dữ liệu và tích hợp AI; vẽ sơ đồ use case, class diagram, sequence diagram.
- **Lập trình và huấn luyện mô hình:** sử dụng Nuxt.js, Vue.js cho frontend, Spring Boot cho backend, Keras/TensorFlow cho mô hình LSTM, Docker để container hóa toàn bộ hệ thống.
- **Triển khai và kiểm thử:** thực hiện kiểm thử chức năng, bảo mật, khả năng tích hợp và độ chính xác của mô hình gợi ý (cosine similarity đạt ~0.945).
- **Kiểm thử, đánh giá và hoàn thiện:** Thực hiện kiểm thử các chức năng, đánh giá hiệu suất, khả năng mở rộng, bảo mật và trải nghiệm người dùng, từ đó hoàn thiện hệ thống.

6. Cấu trúc đồ án:

- **Mở đầu**
- **Chương 1**: Giới thiệu đề tài
- **Chương 2**: Phân tích và thiết kế hệ thống
- **Chương 3**: Triển khai hệ thống
- **Kết luận**

CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI.

1.1. Tổng quan về hệ thống xem phim tích hợp AI gợi ý

Trong bối cảnh công nghệ thông tin ngày càng phát triển mạnh mẽ, nhu cầu giải trí của người dùng cũng đang dần chuyển dịch sang môi trường trực tuyến. Một trong những lĩnh vực có sự chuyển đổi số rõ nét là ngành công nghiệp điện ảnh, đặc biệt là các hệ thống đặt vé xem phim trực tuyến. Thay vì phải đến trực tiếp rạp chiếu phim để mua vé, người dùng hiện nay có thể dễ dàng lựa chọn phim, giờ chiếu, ghế ngồi và thanh toán chỉ với vài thao tác thông qua các ứng dụng web hoặc di động. Điều này không chỉ nâng cao trải nghiệm người dùng mà còn tối ưu hóa vận hành cho các rạp chiếu phim.

Tuy nhiên, việc có quá nhiều phim và suất chiếu khiến người dùng gặp khó khăn trong việc lựa chọn nội dung phù hợp với sở thích cá nhân. Do đó, việc tích hợp các hệ thống **trí tuệ nhân tạo (AI)** nhằm **gợi ý phim cá nhân hóa** đã trở thành một xu hướng tất yếu. Các thuật toán học máy – đặc biệt là mạng nơ-ron hồi tiếp (RNN) và biến thể **LSTM (Long Short-Term Memory)** – tỏ ra hiệu quả trong việc phân tích hành vi người dùng và đề xuất nội dung phù hợp dựa trên lịch sử tương tác.

Hệ thống "Rạp chiếu phim tích hợp AI gợi ý" được xây dựng nhằm giải quyết hai mục tiêu cốt lõi:

- Thứ nhất, **cung cấp một nền tảng đặt vé xem phim hiện đại** với giao diện thân thiện, quy trình đơn giản và bảo mật cao.
- Thứ hai, **áp dụng mô hình học sâu LSTM** để gợi ý phim cho người dùng, dựa trên các đặc trưng như thể loại phim,điểm đánh giá, mức độ phổ biến và nội dung mô tả.

Dữ liệu phim được khai thác từ **TMDB (The Movie Database) API**, đảm bảo tính đa dạng, đầy đủ thông tin về nội dung, diễn viên, trailer và thể loại. Hệ thống được phát triển với kiến trúc microservice:

- Frontend sử dụng **Nuxt.js (Vue 2)**
- Backend phát triển bằng **Spring Boot**
- Server AI triển khai bằng **Flask (Python)**
- Dữ liệu được lưu trữ trên **MySQL và Redis**,
- Toàn bộ hệ thống được container hóa bằng **Docker** và triển khai trên **VPS thực tế**.

Sự kết hợp giữa công nghệ web hiện đại và trí tuệ nhân tạo không chỉ giúp tự động hóa quy trình phục vụ khách hàng mà còn mang lại trải nghiệm giải trí mang tính cá nhân hóa cao, từ đó tăng mức độ hài lòng và giữ chân người dùng.

1.2. Cơ sở lý thuyết

1.2.1. Mạng LSTM (Long Short-Term Memory)

Mạng nơ-ron LSTM (Long Short-Term Memory)[1] là một dạng đặc biệt của Mạng Nơ-ron Hồi quy (Recurrent Neural Network - RNN) được thiết kế để giải quyết vấn đề **phụ thuộc xa (long-term dependencies)** mà RNN truyền thống thường gặp phải. Điều này có nghĩa là LSTM có khả năng ghi nhớ thông tin trong một khoảng thời gian dài, điều rất quan trọng đối với các dữ liệu chuỗi như văn bản, giọng nói, hay chuỗi thời gian.

Mô hình LSTM được đề xuất lần đầu bởi Hochreiter và Schmidhuber vào năm 1997 [3].

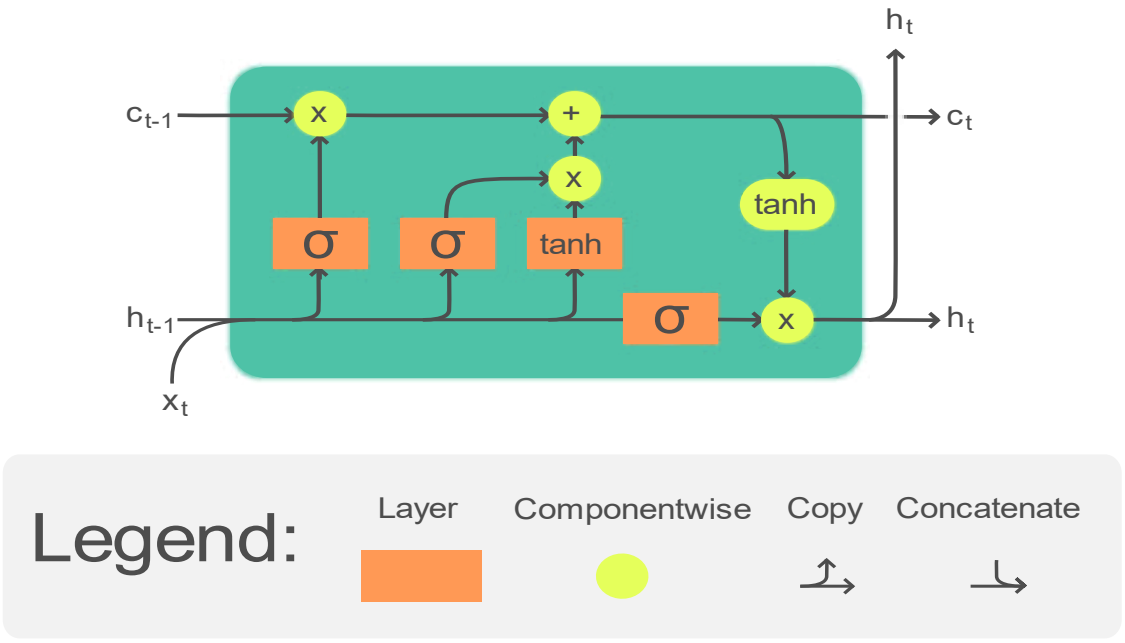
1.2.2.1 Tại sao cần LSTM? Vấn đề của RNN truyền thống

RNN(Recurrent Neural Network)[31] được thiết kế để xử lý dữ liệu chuỗi bằng cách truyền thông tin từ bước thời gian này sang bước thời gian khác thông qua một **trạng thái ẩn (hidden state)** ht. Tuy nhiên, RNN truyền thống gặp phải hai vấn đề chính:

1. **Vấn đề Vanishing Gradient (Gradient biến mất)[32]:** Khi chuỗi dữ liệu dài, gradient (đạo hàm) được lan truyền ngược (backpropagation) qua nhiều bước thời gian sẽ trở nên rất nhỏ, gần như bằng 0. Điều này làm cho các trọng số ở các lớp đầu tiên của mạng không được cập nhật hiệu quả, khiến mạng khó học được các phụ thuộc xa.
2. **Vấn đề Exploding Gradient (Gradient bùng nổ)[33]:** Ngược lại với vanishing gradient, gradient có thể trở nên cực kỳ lớn, dẫn đến việc cập nhật trọng số không ổn định và khiến mô hình "nổ".

LSTM được tạo ra để giải quyết đặc biệt vấn đề vanishing gradient, giúp mạng có thể "ghi nhớ" thông tin quan trọng qua nhiều bước thời gian.

1.2.2.2. Cấu trúc và cơ chế hoạt động của LSTM



Hình 1.1: Kiến trúc mô hình LSTM

Điểm khác biệt cốt lõi của LSTM so với RNN truyền thống là cấu trúc bên trong của mỗi **tế bào LSTM (LSTM cell)**. Thay vì chỉ có một tầng mạng nơ-ron đơn giản, mỗi tế bào LSTM có bốn tầng tương tác với nhau theo một cách đặc biệt, với sự điều khiển của ba **cổng (gates)** và một **trạng thái ô nhớ (cell state)**.

1. Trạng thái ô nhớ (Cell State) - Ct

Đây là "đường dẫn" chính truyền thông tin qua toàn bộ chuỗi. Nó chạy xuyên suốt tế bào, cho phép thông tin truyền đi gần như không thay đổi. Trạng thái ô nhớ có thể được coi là "bộ nhớ dài hạn" của LSTM, lưu trữ thông tin từ các bước thời gian rất xa.

2. Các Cổng (Gates)

Các cổng trong LSTM là các "công tắc" điều khiển luồng thông tin vào, ra và bên trong tế bào. Mỗi cổng là một lớp mạng nơ-ron với hàm kích hoạt sigmoid σ, cho ra giá trị từ 0 đến 1. Giá trị này sẽ nhân (phép nhân Hadamard) với thông tin đi qua cổng, quyết định mức độ thông tin đó được "cho phép" đi qua.

Có ba loại cổng chính:

a. Cổng quên (Forget Gate) - ft

- **Chức năng:** Quyết định thông tin nào từ trạng thái ô nhớ ở bước thời gian trước (C_{t-1}) cần bị "quên" (loại bỏ) và thông tin nào cần được "giữ lại".
- **Công thức:** $f_t=\sigma(W_f\cdot[h_{t-1},x_t]+b_f)$ Trong đó:
    - x_t: Đầu vào hiện tại.
    - h_{t-1}: Trạng thái ẩn (output) của tế bào LSTM ở bước thời gian trước.
    - W_f: Ma trận trọng số của cổng quên.
        - b_f: Vector bias của cổng quên.
        - σ: Hàm sigmoid, tạo ra giá trị trong khoảng [0, 1]. Giá trị 0 nghĩa là "quên hoàn toàn", giá trị 1 nghĩa là "giữ lại hoàn toàn".

b. Cổng đầu vào (Input Gate) - it và Trạng thái ô nhớ ứng cử viên (Candidate Cell State) - C~t

- **Chức năng:** Quyết định thông tin mới nào từ đầu vào hiện tại (x_t) cần được thêm vào trạng thái ô nhớ. Quá trình này bao gồm hai phần:
    - **Cổng đầu vào (it):** Quyết định những giá trị nào sẽ được cập nhật. $i_t=\sigma(W_i\cdot[h_{t-1},x_t]+b_i)$
    - **Trạng thái ô nhớ ứng cử viên ($\tilde{C}_t$):** Tạo ra một vector các giá trị mới tiềm năng để thêm vào trạng thái ô nhớ. Hàm kích hoạt tanh đưa giá trị về khoảng [-1, 1]. $\tilde{C}_t=\tanh(W_C\cdot[h_{t-1},x_t]+b_C)$
- **Cập nhật trạng thái ô nhớ mới (C_t):** $C_t=f_t\odot C_{t-1}+i_t\odot \tilde{C}_t$ Trong đó:
    - $\odot$: Phép nhân Hadamard (element-wise multiplication).
    - $f_t\odot C_{t-1}$: Giữ lại một phần của trạng thái ô nhớ cũ dựa vào quyết định của cổng quên.
    - $i_t\odot \tilde{C}_t$: Thêm thông tin mới vào trạng thái ô nhớ dựa vào quyết định của cổng đầu vào và các giá trị ứng cử viên.

c. Cổng đầu ra (Output Gate) - ot

- **Chức năng:** Quyết định phần nào của trạng thái ô nhớ hiện tại (C_t) sẽ được đưa ra làm **trạng thái ẩn (hidden state)** h_t (và cũng là đầu ra của tế bào LSTM tại bước thời gian đó).
- **Công thức:** $o_t=\sigma(W_o\cdot[h_{t-1},x_t]+b_o)$ $h_t=o_t\odot\tanh(C_t)$ Trong đó:

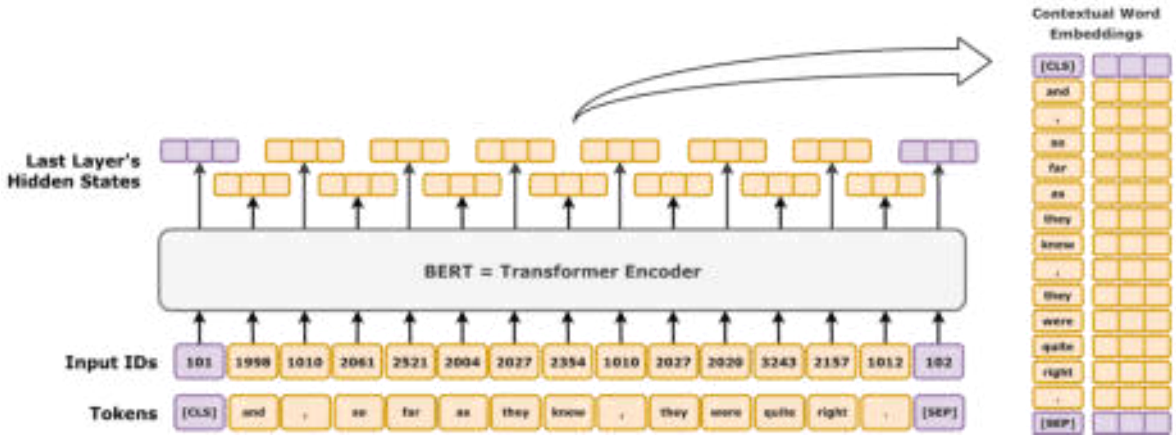
- ht: Trạng thái ẩn mới (output của tế bào LSTM).
- tanh(Ct): Đưa trạng thái ô nhớ về khoảng [-1, 1] trước khi nhân với cổng đầu ra.
- ot⊙tanh(Ct): Lọc thông tin từ trạng thái ô nhớ để tạo ra đầu ra cuối cùng.

Tóm tắt luồng dữ liệu trong một tế bào LSTM:

1. **Cổng quên** quyết định thông tin nào từ Ct−1 sẽ được giữ lại.
2. **Cổng đầu vào** và **Candidate Cell State** quyết định thông tin mới nào từ xt sẽ được thêm vào Ct.
3. **Trạng thái ô nhớ mới** Ct được tính toán từ Ct−1, ft, it và C~t.
4. **Cổng đầu ra** quyết định ht (đầu ra của tế bào) từ Ct.

1.2.2. Embeding văn bản với Bert

BERT (Bidirectional Encoder Representations from Transformers) [2] là một mô hình ngôn ngữ mạnh mẽ được phát triển bởi Google AI vào năm 2018. Điểm nổi bật của BERT so với các mô hình truyền thống là khả năng **hiểu ngữ cảnh theo cả hai chiều (trái → phải và phải → trái)** trong văn bản, nhờ kiến trúc Transformer được thiết kế với các lớp **encoder đa đầu (multi-head attention)**.



Hình 1.2: Bidirectional Encoder Representations from Transformers.

1.2.2.1. Ngữ cảnh hai chiều (Bidirectional context)

Thông thường, các mô hình như RNN, LSTM hoặc GPT chỉ hiểu được ngữ cảnh một chiều (từ trái sang phải hoặc ngược lại). BERT thì khác:

- Khi đọc câu "The movie was not bad at all", BERT không chỉ hiểu từng từ dựa vào từ trước nó, mà còn từ sau nó.
- Điều này giúp mô hình **hiểu rõ hơn về nghĩa thật sự của câu**, ví dụ "not bad" mang nghĩa tích cực thay vì tiêu cực.

1.2.2.2. Token đặc biệt [CLS] và embedding đầu ra

Trong quá trình tiền huấn luyện, BERT sử dụng một token đặc biệt gọi là **[CLS] (Classification)** được chèn vào đầu mỗi câu đầu vào.

- Sau khi đi qua toàn bộ kiến trúc Transformer, **embedding của token [CLS] ở lớp cuối cùng được xem như một vector đại diện toàn bộ câu**.
- Vector này thường có kích thước 768 hoặc 1024 tùy theo phiên bản BERT (Base hay Large).

Đây là cách phổ biến để **biểu diễn embedding ngữ nghĩa của một câu, đoạn văn hoặc mô tả phim**.

1.2.3. Tính Cosine similarity

Cosine Similarity [34] là một độ đo phổ biến trong lĩnh vực khai phá dữ liệu và xử lý ngôn ngữ tự nhiên, được sử dụng để đánh giá mức độ tương đồng giữa hai vector trong không gian vector. Khác với khoảng cách Euclidean đo khoảng cách tuyệt đối, Cosine Similarity đánh giá độ giống nhau dựa trên **góc giữa hai vector**, không quan tâm đến độ lớn của chúng. Do đó, độ tương đồng cosine đặc biệt hiệu quả trong các bài toán mà độ dài văn bản không quan trọng, như khi so sánh nội dung văn bản, mô tả phim, v.v.

Công thức tính Cosine Similarity giữa hai vector AAA và BBB được xác định như sau:

$$\text{CosineSimilarity}(A, B) = \frac{A \cdot B}{\|A\| \cdot \|B\|} = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Hình 1.3: Công thức tính CosineSimilarity.

Trong đó:

- A·B là tích vô hướng của hai vector.
- ‖A‖,‖B‖ là độ dài (chuẩn Euclidean) của hai vector.

Giá trị Cosine Similarity nằm trong khoảng từ -1 đến 1, trong đó:

- Giá trị 1 thể hiện hai vector cùng hướng (giống nhau hoàn toàn),
- Giá trị 0 thể hiện hai vector vuông góc (không liên quan),
- Giá trị -1 thể hiện hai vector ngược hướng (tương phản hoàn toàn).

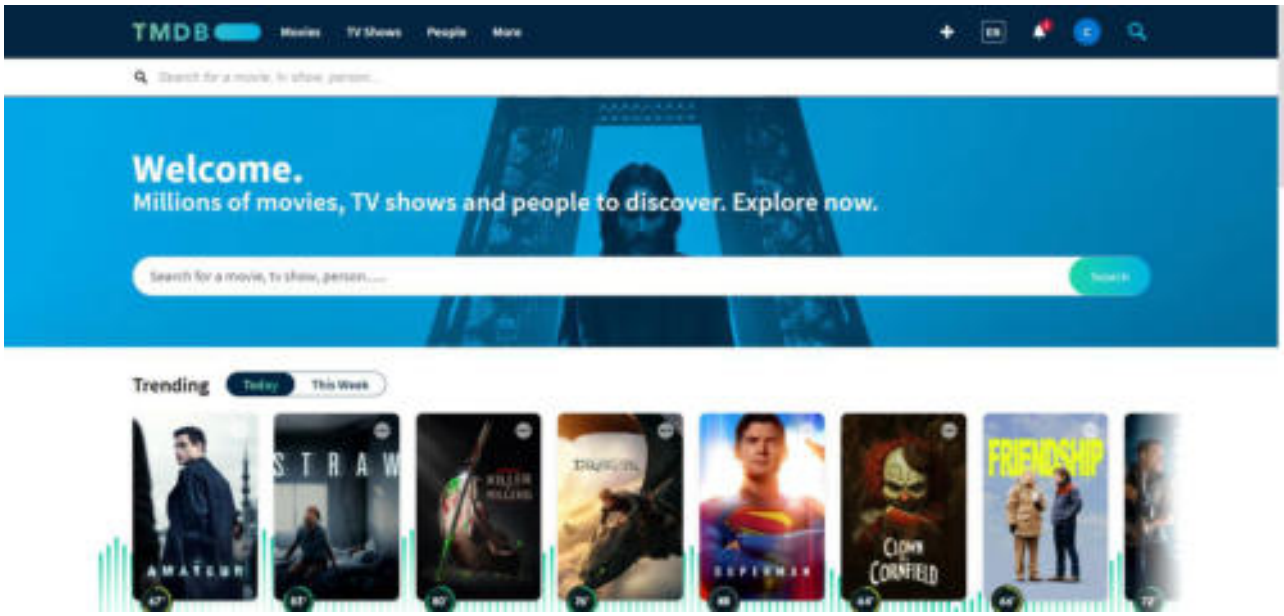
Tuy nhiên, trong thực tế các ứng dụng xử lý văn bản, giá trị cosine thường nằm trong khoảng từ 0 đến 1 do các vector thường không mang giá trị âm.

Trong hệ thống gợi ý phim, mỗi phim có thể được biểu diễn bởi một vector đặc trưng (embedding) thu được từ mô tả nội dung phim thông qua các mô hình ngôn ngữ như BERT. Cosine Similarity được sử dụng để đo mức độ tương đồng ngữ nghĩa giữa các phim, hoặc giữa sở thích người dùng và nội dung phim. Phim nào có độ tương đồng cao hơn sẽ được ưu tiên đề xuất.

1.2.4. TMDB DataSet & TMDB API (nền tảng dự án)

1.2.4.1. Giới thiệu về TMDB

- TMDB (The Movie Database) [1] là cơ sở dữ liệu điện ảnh trực tuyến lớn, khởi nguồn từ năm 2008, chứa thông tin phong phú về phim, diễn viên, điểm số, thể loại, mô tả, ngày phát hành…
- Dữ liệu được đóng góp bởi cộng đồng toàn cầu, hỗ trợ đa ngôn ngữ và được duy trì chất lượng bởi các moderator.



Hình 1.4: TMDB website.

1.2.4.2. TMDB 100000 Movie Metadata Dataset

Là một tập dữ liệu phổ biến dùng trong các dự án gợi ý phim demo, chứa ~100.000 phim từ TMDB, phù hợp với việc huấn luyện mô hình recommender đã triển khai

Thông tin chính bao gồm: id, title, overview, genres, vote_average, popularity, và nhiều trường metadata khác như keywords, cast, crew

1.2.4.3. TMDB API – API v3/v4

API TMDB cho phép:

- Tìm kiếm phim (/search/movie)
- Lấy chi tiết phim bằng movie_id (/movie/{movie_id})
- Lấy poster, poster_path, và hình ảnh khác (/movie/{movie_id}/images)
- Các endpoint khác như credit, similar và popular movies.

Cần đăng ký API key, tuân thủ hạn mức truy vấn và điều khoản sử dụng

developer.

1.2.5. Lý thuyết nền tảng Docker

Docker [14] là nền tảng ảo hóa cấp thấp (containerization) cho phép đóng gói ứng dụng và toàn bộ phụ thuộc (runtime, thư viện, cấu hình...) vào một container nhẹ, chia sẻ chung nhân hệ điều hành nhưng vẫn tách biệt môi trường chạy του mỗi component .

Ưu điểm chính:

- Nhẹ và nhanh: Container khởi động nhanh hơn nhiều so với máy ảo, tiêu tốn tài nguyên nhỏ hơn nhờ chia sẻ kernel.

- Tính nhất quán môi trường: Docker đảm bảo ứng dụng chạy giống y hệt ở development, testing và production, giải quyết triệt để lỗi "chạy được trên máy mình".

- Di động: Container có thể chạy trên mọi máy có Docker Engine (kể cả Linux, Windows, macOS).

- Tính microservice: Mặc định mỗi service (backend, frontend, AI service, Redis, MySQL…) chạy trong container riêng, dễ quản lý và mở rộng .

1.2.5.1. Docker cho Spring Boot backend

- Container hóa ứng dụng Spring Boot bằng Docker giúp xây dựng "fat-jar" sẵn kèm runtime, JDBC driver… và chạy độc lập trong môi trường ổn định.

- Việc ứng dụng Docker với Spring Boot giúp giảm thời gian deploy, cải thiện startup time và resource isolation.
- Sử dụng best practice như: multi-stage build, cụ thể hóa base image và tối ưu hóa kích thước image (vd: eclipse-temurin:17-jdk-jammy).

1.2.5.2. Docker cho NuxtJs frontend

- Đảm bảo nhất quán môi trường: Docker truy xuất Node.js, npm/yarn và các phụ thuộc giống hệt từ development đến production – tránh lỗi do khác Node phiên bản.

- Khởi động nhanh và nhẹ: Container Vue/Nuxt thường chứa chỉ mã chạy (không phải toàn bộ dev tools), khởi chạy nhanh và chiếm ít tài nguyên hơn máy ảo .

- Hỗ trợ scale ngang dễ dàng khi cần chạy nhiều instance frontend.

1.2.5.3. Docker hóa Flask AI Service & Redis Cache

- Theo phương pháp phổ biến, AI service (Flask[16]) và Redis cache được đặt trong các container riêng biệt, kết nối qua mạng riêng giữa các service.

- Thiết kế này giúp service AI và Redis có thể bảo trì, scale lên/xuống độc lập – rất phù hợp với hệ kiến trúc microservice.

1.2.6. Docker Compose – Orchestration đa Container

Docker Compose [15] là công cụ để định nghĩa và chạy multi-container environment bằng file docker-compose.yml:

- Cho phép cấu hình container như build context, volumes, network, ports, dependencies một cách dễ dàng .
- Trong dự án "Cinema System", một file yml chứa các service frontend, backend, AI service, Redis, MySQL giúp:
    - Khởi động đồng loạt bằng docker-compose up.
    - Liên kết service tự động (DNS nội container theo tên service).
    - Sử dụng **volumes** để lưu dữ liệu MySQL/Redis và code reload khi dev.

1.2.7. Ứng dụng thực tế & CI/CD

- Docker giúp dễ dàng unit-test, integration-test, và deploy bằng CI/CD, như GitHub Actions [28] , Jenkins – đóng vai trò trong dự án gợi ý phim.

- Quick spin-up/down environment khi cần chạy thử, debug hoặc trước khi release production – đảm bảo tính ổn định và tốc độ deploy .

Bảng 1.1: Docker & Docker Compose.

Mục đích	Công cụ	Tác dụng
Đóng gói / environment	Docker	Container hóa Spring Boot, Flask AI service, Redis, MySQL để deploy nhất quán
Quản lý nhiều service đồng thời	Docker Compose	Orchestration dev/test/prod với một cấu hình chung
Frontend container hóa	Docker	Đóng gói Nuxt 2 + Node.js để build và chạy ổn định trên mọi môi trường
Isolation & consistency	Docker	Tách biệt service, tránh xung đột dependencies
Tăng tốc deploy & tích hợp CI/CD	Docker	Dễ dàng tích hợp với GitHub Actions, Jenkins, deploy nhanh, rollback an toàn

1.3. Công nghệ sử dụng trong dự án

1.3.1. Công nghệ Front-End

1.3.1.1. Tổng quan Nuxt 2 (Vue2)

- Nuxt.js (phiên bản 2) [20] là framework phát triển từ Vue 2, cung cấp môi trường Server-Side Rendering (SSR), đồng thời hỗ trợ chế độ SPA/SSG, giúp cải thiện SEO và tốc độ tải trang đầu tiên
- Nuxt hỗ trợ zero-config, module hệ thống, và các best practices mặc định, giúp phát triển frontend quy mô lớn trở nên dễ dàng hơn mà không mất nhiều thời gian cấu hình

1.3.1.2. Các tính năng nổi bật

Bảng 1.2: Các tính năng nổi bật.

Tính năng	Mô tả
Server-Side Rendering	Trang được dựng sẵn trên server và trả về HTML hoàn chỉnh, giúp cải thiện tốc độ hiển thị và SEO .
File-based Routing	Tự động tạo route dựa trên cấu trúc thư mục pages/, giúp giảm thiểu cấu hình router thủ công .
Auto-import Components	Tự động import component từ thư mục components/, giúp frontend gọn gàng và bảo trì dễ hơn .
Code Splitting & Prefetch	Nuxt tự động chia nhỏ bundle và prefetch khi người dùng tương tác: tăng tốc độ load cho các trang sau .
Layouts, Middleware	Hỗ trợ layouts/route middleware cho giao diện và xác thực (auth guards), giúp tổ chức code rõ ràng hơn.
Vuex	Quản lý trạng thái ứng dụng (user, booking, token) một cách hiệu quả.
Progressive Enhancement	Kết hợp SSR + client-side hydration giúp trải nghiệm mượt mà tương tự SPA khi trang đã tải .

1.3.1.3. Lợi ích trong dự án

- Tăng hiệu suất & SEO: Các trang phim và trang lịch sử được render sẵn giúp load nhanh, hỗ trợ bots của Google dễ crawl nội dung .
- Phát triển nhanh & có cấu trúc: Tính năng auto-import, routing file-based, module hệ thống giúp code dự án "Cinema System" sạch và nhất quán.
- Khả năng mở rộng & bảo trì: Dễ dàng thêm module, plugin, hoặc layout mới mà không disrupt codebase hiện tại.
- UX mượt mà: Sau lần tải đầu, các tương tác tiếp theo như chọn ghế, đặt vé đều giống SPA nhưng không mất SEO.

1.3.1.4. Mô hình triển khai

- Frontend thực hiện SSR tại thời điểm npm run build && npm run start trên Docker container, và tiếp tục hoạt động dưới dạng hydrated Vue SPA để xử lý các tương tác người dùng (NuxtLink, store, fetch).
- Client-Server flow:
  - Trình duyệt gửi yêu cầu → server render HTML + fetch data (asyncData/fetch).
  - Đường dẫn NuxtLink chuyển trang SPA mà không reload toàn bộ.

- Vue components tự động import và reactivity được khởi chạy – pages trở nên reactive.

1.3.2. Công nghệ Back-end Java SpringBoot

1.3.2.1. Tổng quan công nghệ

- Spring Boot [11] là framework giúp xây dựng nhanh ứng dụng Java gọn nhẹ, không cần cấu hình phức tạp, đi kèm server nhúng (Tomcat, Jetty) để chạy standalone ứng dụng trong production
- Dự án sử dụng Spring Boot để phát triển **RESTful API**, xử lý nghiệp vụ như đăng ký, đăng nhập, đặt vé, lấy gợi ý qua móđun recommendation.

1.3.2.2.Các thành phần chính

Bảng 1.3: Các thành phần chính Java SpringBoot.

Thành phần	Công nghệ / Library	Vai trò
Web & REST API	spring-boot-starter-web	Cung cấp DispatcherServlet, JSON serialization, REST endpoint.
Data Persistence	Spring Data JPA + Hibernate + MySQL Driver	Kết nối ORM với MySQL để quản lý người dùng và booking.
Authentication	Spring Security + JWT	Bảo vệ endpoint, xác thực và phân quyền người dùng qua token JWT.
Actuator & Config	spring-boot-starter-actuator, externalized config	Giám sát, healthcheck, metrics, cấu hình linh hoạt .
Maven	Quản lý build & dependencies	Sử dụng Maven để build, chạy tests và quản lý dependencies.

1.3.2.3. Bảo mật & JWT

- Spring Security [12] kết hợp với JWT để xác thực theo kiểu stateless: người dùng đăng nhập → token được cấp → gửi kèm trong header Authorization cho các API lúc sau.
- Backend sử dụng filter hoặc cấu hình OAuth2 Resource Server để validate token và phân quyền theo claim bên trong token.

1.3.2.4. Kết nối Database

- Sử dụng Spring Data JPA + Hibernate cho phép viết Entity class và Repository interface, tự động tạo bảng và CRUD operation.
- Kết nối đến MySQL định kỳ, đảm bảo lưu lại thông tin người dùng, vé đã đặt, hệ thống có thể mở rộng khi cần.

1.3.2.5. Công nghệ Container & Docker

- Backend được Docker hóa với một Dockerfile sử dụng JDK image, copy artifact JAR và chạy dưới mục CMD ["java", "-jar", …].

- Kết hợp với Docker Compose, Backend được deploy cùng các container khác (Redis, MySQL, AI service…) giúp hệ thống chạy ổn định trên VPS Ubuntu

1.3.2.6. Lý do chọn Spring Boot

- Nhanh chóng & ít boilerplate code, giảm thiểu cấu hình thủ công .
- Dễ bảo mật với Spring Security và JWT, phù hợp với API REST stateless.
- Dễ scale và tích hợp với Docker cũng như CI/CD pipelines.
- Giàu tính năng production-ready, như Actuator và externalized config, phù hợp mô hình microservices.

1.3.3. AI Service – Flask Server (Python)

1.3.3.1. Khái niệm & vai trò

Flask [16] là một micro-framework nhẹ cho Python, thiết kế để xây dựng nhanh chóng các dịch vụ web nhỏ gọn và RESTful . Trong dự án "Cinema System", Flask được dùng làm AI service độc lập, phục vụ mô hình LSTM để gợi ý phim thông qua API endpoint /api/recommendations.

1.3.3.2. Ưu điểm nổi bật

- Đơn giản & tập trung: Chỉ cần vài tập tin cấu hình để expose mô hình dưới dạng REST API, phù hợp với dự án nhỏ và microservice .
- Kết nối tự nhiên với AI/ML: Flask tích hợp trực tiếp với các thư viện Keras [4], TensorFlow [5], NumPy,… thuận tiện để load và thực thi mô hình .
- Triển khai nhẹ nhàng trên Docker: Container hóa dễ dàng, kết hợp tốt với Redis để cache embedding hoặc kết quả gợi ý, đáp ứng pattern microservice .

1.3.3.3. Tích hợp với mô hình LSTM & Redis

- Sau khi huấn luyện xong, mô hình LSTM được lưu file .h5 và load khi service khởi động.
- Khi endpoint nhận request (danh sách movie IDs), Flask trả về list phim gợi ý bằng cách apply mô hình.
- Để tối ưu tốc độ, service có thể cache kết quả hoặc embedding trong Redis: khi request lặp lại, tránh phải tính toán lại .

1.3.3.4. Triển khai và vận hành

- Mỗi dịch vụ Flask chạy standalone dưới Python, dễ dàng deploy trong Docker với thư viện cần thiết.
- Khi kết hợp Docker Compose (cùng backend, frontend, redis, mysql), Flask đóng vai trò như một microservice độc lập, có thể scale và giám sát riêng biệt .

1.3.3.5. Vì sao chọn Flask

Bảng 1.4: Lí do chọn Flask.

Ưu điểm	Giải thích
Micro-framework	Phù hợp khi chỉ cần triển khai mô hình LSTM, không cần MVC toàn diện như Django
Hiệu quả với ML	Kết nối trực tiếp model Keras, dễ chuẩn bị input/output JSON
Tích hợp Docker tốt	Container nhẹ, dễ deploy cùng CI/CD pipeline
Tăng tốc qua cache	Redis hỗ trợ phục vụ gợi ý nhanh mà không re-train

1.3.4. Cơ sở dữ liệu MYSQL

1.3.4.1. Tổng quan

MySQL [26] là hệ quản trị cơ sở dữ liệu quan hệ (Relational Database Management System - RDBMS) mã nguồn mở, được phát triển bởi Oracle. Trong dự án "Cinema System", MySQL được sử dụng làm cơ sở dữ liệu chính, để lưu trữ thông tin người dùng, phim, suất chiếu, và lịch sử đặt vé.

1.3.4.2. Vai trò trong hệ thống

MySQL giúp lưu trữ dữ liệu có cấu trúc và quan hệ rõ ràng thông qua các bảng, khóa ngoại, ràng buộc dữ liệu, đảm bảo toàn vẹn dữ liệu và khả năng truy vấn nhanh chóng. Hệ thống sử dụng Spring Boot để tương tác với MySQL thông qua Spring Data JPA và Hibernate.

1.3.4.3. Các đặc điểm nổi bật

- **Hỗ trợ ACID**: Đảm bảo giao dịch an toàn, nhất quán và không bị mất dữ liệu trong trường hợp lỗi.
- **Khả năng mở rộng**: Có thể hoạt động tốt trên nhiều quy mô từ nhỏ đến lớn.
- **Tích hợp tự động** với Spring Boot: Tự sinh schema khi khởi động, hỗ trợ truy vấn linh hoạt qua Repository interface.
- **Hỗ trợ Docker hoàn hảo**: MySQL chạy trong container, dễ deploy, dễ backup restore và chia sẻ cùng network với các container khác.

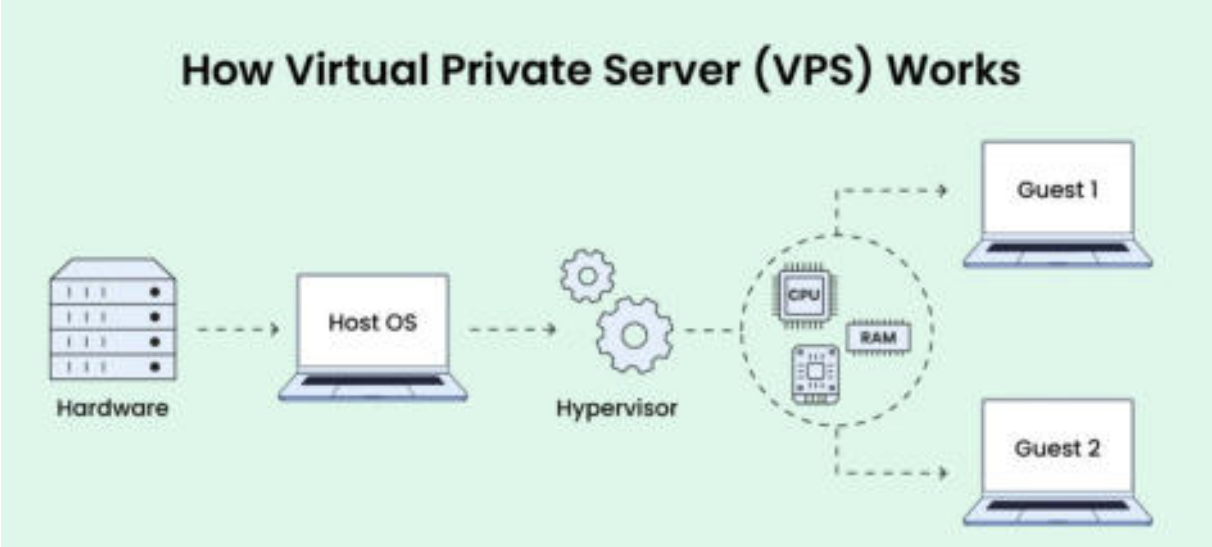
1.3.4.4. Cách sử dụng trong dự án

- MySQL được khởi tạo qua Docker Compose, được mount volume để lưu trữ dữ liệu lâu dài.
- Spring Boot kết nối qua JDBC với thông tin khai báo trong application.properties.
- Dữ liệu được định nghĩa bằng Entity class, quan hệ được xác định qua JPA annotation.
- Toàn bộ dữ liệu phục vụ cho API backend và gới ý phim.

1.3.5. Hạ tầng triển khai

1.3.5.1. VPS

VPS (Virtual Private Server) [35] là một máy chủ ảo được tạo ra bằng cách phân chia một máy chủ vật lý thành nhiều máy chủ ảo. Mỗi VPS hoạt động như một máy chủ độc lập với hệ điều hành riêng, tài nguyên CPU, RAM, và ổ cứng được phân bổ cụ thể. Đây là một giải pháp lưu trữ trung gian giữa shared hosting (lưu trữ chia sẻ) và dedicated server (máy chủ riêng).



Bảng 1.5: Sơ đồ cấu trúc VPS.

VPS hoạt động dựa trên công nghệ ảo hóa, cho phép một máy chủ vật lý (physical server) được chia thành nhiều máy chủ ảo độc lập. Dưới đây là cách nó hoạt động:
Một phần mềm ảo hóa (hypervisor) như VMware, Hyper-V hoặc KVM được cài trên máy chủ vật lý. Hypervisor phân chia tài nguyên (CPU, RAM, ổ cứng, băng thông) thành các phần riêng biệt. Mỗi VPS được cấp một phần tài nguyên cố định hoặc linh hoạt, hoạt động như một máy chủ riêng với hệ điều hành (OS) riêng (Windows, Linux, v.v.), không bị ảnh hưởng bởi các VPS khác. Mỗi VPS có môi trường riêng, bao gồm địa chỉ IP, cấu hình mạng, và bộ nhớ, đảm bảo an toàn và không bị can thiệp từ các VPS khác trên cùng máy chủ vật lý. Người dùng có quyền truy cập root/admin để cài đặt phần mềm, quản lý dữ liệu và cấu hình theo nhu cầu, giống như một máy chủ chuyên dụng, nhưng với chi phí thấp hơn. Nhờ chia sẻ phần cứng vật lý, VPS tận dụng hiệu quả tài nguyên, cho phép mở rộng linh hoạt khi nhu cầu tăng (scaling) mà không cần thay đổi phần cứng.

Nguyên lý này giúp VPS cung cấp hiệu suất ổn định, bảo mật cao và phù hợp cho website, ứng dụng, hoặc dịch vụ cần tài nguyên riêng biệt với chi phí hợp lý.

1.3.5.2. Đặc điểm của VPS

- **Toàn quyền root/sudo**: Tự do cài đặt các dịch vụ, cấu hình firewall, deploy code, backup.
- **IP riêng**: Giúp dễ truy cập từ xa, hỗ trợ mapping domain, cấp SSL, mốc API.
- **Tự do deploy**: Hỗ trợ cài Docker, Node.js, Java, Python, MySQL, Redis... đồng thời và linh hoạt.
- **Dễ backup & restore**: Nhiều VPS có tính năng snapshot, backup theo lịch.

1.3.5.2. VPS vs Shared Hosting

Bảng 1.6: Bảng so sánh VPS và Shared Hosting.

Tiêu chí	***VPS***	***Shared Hosting***
Tài nguyên	Đầy đủ (CPU, RAM, IP)	Chia sẻ chung nhiều website
Quản lý hệ điều hành	Có (SSH root)	Bị giới hạn, theo control panel
Tự do deploy	Cao	Hạn chế nghiêm ngặt
Bảo mật	Tốt hơn (có firewall riêng)	Rủi ro cao (nhiều user chia)

1.3.6. Vpssieutoc.vn

Hình 1.5: vpssieutoc.vn.

vpssieutoc.vn là nhà cung cấp VPS tại Việt Nam, hỗ trợ nhiều dòng VPS Linux/Windows với các OS Ubuntu, CentOS, AlmaLinux, v.v. Giao diện quản lý VPS trực quan, cho phép backup, reset, thay mật khẩu root nhanh chóng.

1.3.6.1. Lý do chọn vpssieutoc.vn

- **Chi phí thấp**: Gói VPS cơ bản rất phù hợp với sinh viên hoặc dự án demo.
- **Cố định IP**: Dễ truy cập và deploy API / giao diện web.
- **Chạy ổn định**: Hỗ trợ Docker Compose tốt, chạy đồng thời 4–5 container.
    - **Phụ hợp mô hình microservice**: Kết hợp backend, frontend, AI Flask, Redis, MySQL đồng bộ.

VPS là giải pháp linh hoạt, tiết kiệm chi phí và toàn quyền triển khai hệ thống. vpssieutoc.vn đã cung cấp hạ tầng đáng tin cậy và dễ dùng cho dự án "Cinema System" với cấu hình hợp lý, hỗ trợ đầy đủ Docker, SSH và truy cập từ xa.

1.3.7. Phương thức xác nhận email SMTP

1.3.7.1. Tổng quan lý thuyết về SMTP

SMTP (Simple Mail Transfer Protocol) là giao thức tiêu chuẩn dùng để gửi email giữa các máy chủ. Nó hoạt động trên nền giao thức TCP/IP, thường sử dụng cổng **587 (TLS)** hoặc **465 (SSL)** để truyền tải dữ liệu một cách an toàn.

Trong mô hình 3 lớp gửi mail, SMTP thường đảm nhận vai trò:

- Client → SMTP Server: gửi email từ ứng dụng đến máy chủ gửi thư.
- **SMTP Server → Mail Server đích**: truyền tiếp email đến hòm thư người nhận.

Đặc điểm chính:

- **Một chiều**: SMTP chỉ gửi email, không dùng để nhận (POP3/IMAP mới dùng để nhận).
- **Hoạt động theo mô hình push**: client chủ động đẩy email đến máy chủ.
- **Dễ tích hợp**: thư viện phổ biến trong các ngôn ngữ như Java, Python, PHP đều hỗ trợ SMTP.
- **Phổ biến**: Gmail, Outlook, Zoho Mail, Mailgun... đều hỗ trợ giao thức này.

Ví dụ thông số cấu hình SMTP Gmail:

Bảng 1.7: Thông số cấu hình SMTP Gmail.

Thông số	Giá trị
SMTP Server	smtp.gmail.com
Cổng (TLS)	587
Cổng (SSL)	465
Xác thực	Có (tài khoản Gmail + app password)
Bảo mật	TLS hoặc SSL

Trong dự án này, hệ thống sử dụng JavaMailSender của Spring Boot để kết nối tới máy chủ Gmail và gửi email xác nhận tài khoản người dùng thông qua SMTP.

1.3.8. Cơ chế xác thực email qua STMP trong hệ thống

1.3.8.1. Mục đích

Trong hệ thống đặt vé xem phim Cinema System, việc xác thực tài khoản khi người dùng đăng ký là bắt buộc nhằm đảm bảo tính bảo mật và ngăn chặn các tài khoản ảo. Cơ chế này được hiện thực hóa thông qua việc gửi mã xác nhận qua email, sử dụng giao thức SMTP (Simple Mail Transfer Protocol).

1.3.8.2. Cách hoạt động

- Khi người dùng gửi yêu cầu đăng ký, hệ thống backend sinh một mã xác nhận ngẫu nhiên gồm 6 chữ số.
- Mã này được lưu tạm thời vào Redis với thời gian sống (TTL) là 10 phút.
- Đồng thời, hệ thống sử dụng thư viện JavaMailSender trong Spring Boot để gửi email chứa mã xác nhận đến địa chỉ người dùng qua Gmail SMTP.
- Người dùng nhập lại mã có dạng 123456 vào hệ thống.
- Backend kiểm tra mã đang lưu trong Redis. Nếu hợp lệ và chưa hết hạn, tài khoản sẽ được kích hoạt. Redis sau đó sẽ xóa mã xác nhận đã dùng.

1.3.9. Kết hợp Redis và SMTP

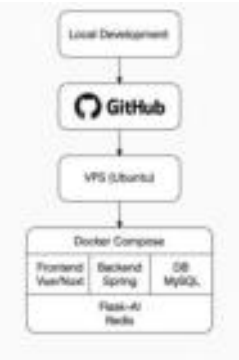
- Redis đóng vai trò như một hệ thống lưu tạm thời, nhẹ, giúp giảm tải truy vấn vào cơ sở dữ liệu chính (MySQL).
- TTL giới hạn giúp đảm bảo tính bảo mật: mã xác nhận chỉ hợp lệ trong một khoảng thời gian nhất định.
- SMTP (cụ thể là Gmail SMTP) cung cấp kênh truyền email tin cậy, không yêu cầu hạ tầng email riêng biệt, phù hợp với môi trường phát triển sinh viên hoặc các hệ thống nhỏ.

1.3.9.1. Lý do chọn giải pháp

- ***Dễ triển khai:*** Gmail SMTP dễ tích hợp, không yêu cầu máy chủ thư chuyên biệt.
- ***Bảo mật cao:*** Mã được lưu ngắn hạn, tự động xóa khỏi Redis sau khi xác minh thành công.
- ***Hiệu năng tốt:*** Redis truy xuất nhanh, phù hợp cho các tác vụ xác thực tạm thời.

1.3.10. Công cụ hỗ trợ triển khai và phát triển dự án

Quá trình phát triển và triển khai hệ thống rạp chiếu phim tích hợp AI gợi ý đã sử dụng nhiều công cụ và nền tảng hỗ trợ một cách hiệu quả. Trong đó, hai công cụ chủ đạo là GitHub và Docker, kết hợp với một số công cụ khác đã giúp quy trình từ phát triển đến triển khai diễn ra nhanh chóng, đồng bộ, và ổn định.



Hình 1.6: Sơ đồ quy trình phát triển & triển khai.

1.3.10.1. GitHub

1.3.10.1.1. Giới thiệu

GitHub [27] là một nền tảng quản lý mã nguồn phân tán được xây dựng trên hệ thống kiểm soát phiên bản **Git**. Công cụ này cho phép nhiều lập trình viên cùng cộng tác trên một dự án, dễ dàng theo dõi lịch sử thay đổi, tạo nhánh (branch), xử lý yêu cầu kéo (pull request), và kiểm soát chất lượng mã thông qua review và CI/CD. Ngoài ra, GitHub còn đóng vai trò trung tâm trong việc đồng bộ hóa giữa môi trường phát triển cục bộ và môi trường triển khai (production), giúp quy trình deploy trở nên khoa học và dễ kiểm soát hơn.

1.3.10.1.2. Cách sử dụng trong dự án

- Là kho lưu trữ chung của toàn bộ source code frontend, backend, AI.
- Phát triển local xong thì commit, push code lên GitHub.
- Trên VPS, dùng git pull để đồng bộ mã nguồn với bản mới nhất.
- Tạo pipeline triển khai CI/CD (tùy chọn).

1.3.10.2. Docker

1.3.10.2.1. Giới thiệu

Docker là nền tảng container hóa giúp đóng gói ứng dụng cùng toàn bộ môi trường chạy (dependencies, runtime, thư viện, cấu hình) vào các **container**. Điều này đảm bảo rằng ứng dụng chạy nhất quán trên mọi môi trường – từ máy lập trình viên đến máy chủ thật (VPS). Ngoài ra, Docker còn hỗ trợ **Docker Compose**, cho phép định nghĩa và quản lý nhiều dịch vụ (frontend, backend, Redis, database...) chỉ trong một tệp cấu hình, đơn giản hóa việc chạy hệ thống nhiều thành phần.

> Việc sử dụng Docker giúp hệ thống tránh được các lỗi xung đột môi trường, tăng tốc quá trình triển khai, đồng thời sẵn sàng cho các bước nâng cấp quy mô trong tương lai như tích hợp CI/CD hoặc triển khai lên Kubernetes.

1.3.10.2.2. Cách sử dụng trong dự án

- Container hóa toàn bộ frontend (Nuxt.js), backend (Spring Boot), AI Flask, Redis, MySQL.
- Dùng Docker Compose quản lý nhiều service trong một file docker-compose.yml.
- Build, run, restart nhanh chóng và dễ scale.

1.3.10.3. Các công cụ khác

- Postman: Kiểm thử API backend nhanh chóng.
- MySQL Workbench / DBeaver: Quản lý cơ sở dữ liệu MySQL.
- Redis CLI: Truy vấn cache mã xác nhận email.
- Visual Studio Code / IntelliJ IDEA: Soạn thảo và debug mã nguồn frontend/backend.
- Google Colab: Huấn luyện mô hình AI với GPU miễn phí.

1.3.11. Hệ thống gợi ý phim thông minh

1.3.11.1. Giới thiệu và lý do áp dụng AI

Trong hệ thống đặt vé xem phim trực tuyến *Cinema System*, việc cá nhân hóa trải nghiệm người dùng là một yếu tố quan trọng để nâng cao mức độ hài lòng và giữ chân khách hàng. Do đó, em đã xây dựng một **mô hình học sâu dựa trên LSTM (Long Short-Term Memory)** nhằm gợi ý các bộ phim phù hợp cho từng người dùng, dựa trên lịch sử đặt vé, thể loại phim yêu thích và mô tả nội dung phim.

1.3.12. Định hướng kiến trúc và mô hình triển khai của hệ thống

Trong quá trình phát triển hệ thống rạp chiếu phim, lựa chọn kiến trúc hiện đại kết hợp giữa **microservices**, **RESTful API** và **container hóa** đã đảm bảo tính linh hoạt, dễ mở rộng, dễ triển khai và bảo trì.

1.3.12.1. Kiến trúc phần mềm: RESTful Microservices

Hệ thống được thiết kế theo mô hình RESTful API, trong đó các thành phần phần mềm giao tiếp với nhau thông qua HTTP protocol và trao đổi dữ liệu dưới định dạng JSON. Đặc điểm chính của mô hình này:

- Client-server: frontend (Nuxt.js) hoàn toàn tách biệt với backend (Spring Boot).
- Stateless: mỗi request là độc lập, không lưu trạng thái phiên ở server.
- Giao tiếp qua HTTP verbs: GET, POST, PUT, DELETE...
- Phản hồi dữ liệu dạng JSON để dễ xử lý phía frontend hoặc AI.

Cấu trúc RESTful giúp frontend linh hoạt trong việc gọi dữ liệu, đồng thời giúp backend mở rộng dễ dàng sang mobile app hoặc tích hợp bên thứ ba.

1.3.12.2. Kiến trúc triển khai: Microservices container hóa

Toàn bộ hệ thống được phân tách thành nhiều thành phần độc lập:

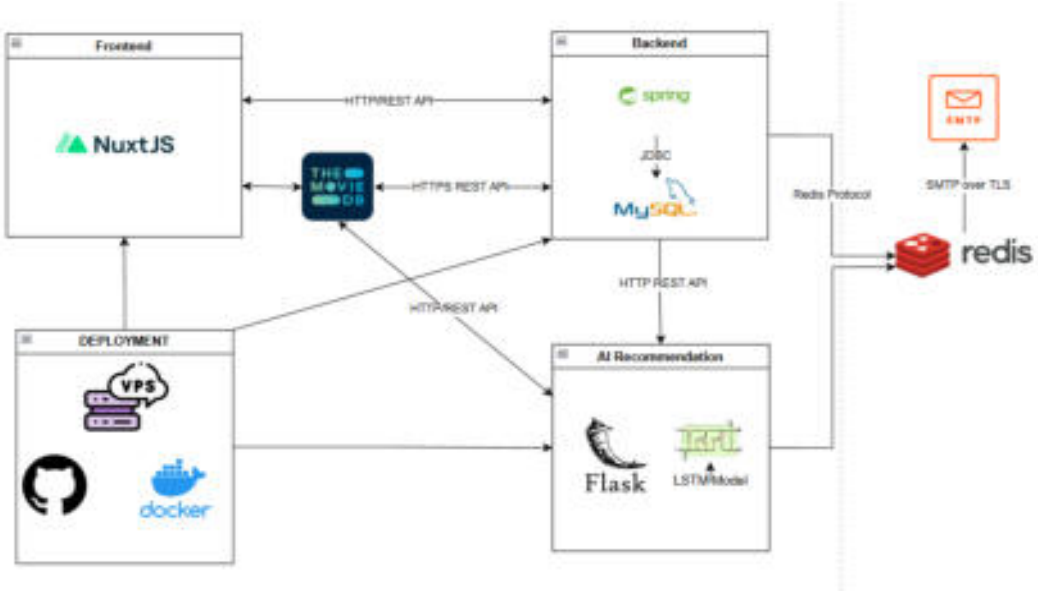
- **Frontend (Nuxt.js – Vue 2):** giao diện người dùng, gọi API backend qua Axios.
- **Backend (Spring Boot):** cung cấp API RESTful cho người dùng (đăng ký, đăng nhập, đặt vé…), xác thực JWT, xử lý email SMTP, và gọi dịch vụ gợi ý phim.
- **Flask server (Python):** cung cấp AI gợi ý phim sử dụng mô hình LSTM.
- **Redis:** lưu cache mã xác nhận email, hoạt động như một kho tạm thời hiệu năng cao.
- **MySQL:** lưu thông tin người dùng, vé, và phim.
- **TMDB API:** lấy dữ liệu phim từ nguồn bên ngoài để phục vụ hiển thị và huấn luyện AI.

Các service trên được triển khai theo dạng **microservices**, nghĩa là mỗi thành phần chạy độc lập trong một container riêng, quản lý tập trung bằng **Docker Compose**. Điều này giúp dễ dàng quản lý, scale độc lập và đồng bộ hóa môi trường trên mọi máy chủ.

1.3.12.3. Mô hình triển khai

Toàn bộ hệ thống được triển khai trên một máy chủ ảo VPS (vpssieutoc.vn) chạy Ubuntu. Mã nguồn được đồng bộ qua GitHub, sau đó pull về máy chủ qua SSH. Tại đây, dự án được dựng và khởi chạy bằng Docker Compose, với cấu hình môi trường riêng biệt cho mỗi container.

1.3.12.4. Sơ đồ tổng quan



Hình 1.7: Sơ đồ kiến trúc hệ thống.

1.4. Tổng kết chương

Chương 1 đã trình bày tổng quan về đề tài "Hệ thống rạp chiếu phim tích hợp AI gợi ý phim", từ cơ sở thực tiễn và nhu cầu xây dựng một nền tảng đặt vé xem phim trực tuyến thông minh, cho đến mục tiêu và phạm vi cụ thể của đồ án. Các chức năng chính của hệ thống, đối tượng người dùng và hướng tiếp cận công nghệ đều đã được xác định rõ ràng.

Bên cạnh đó, chương cũng đưa ra định hướng kiến trúc tổng thể của hệ thống với mô hình microservices kết hợp RESTful API, triển khai thông qua Docker trên nền VPS. Các thành phần như frontend (Nuxt.js), backend (Spring Boot), AI recommendation server (Flask), Redis, MySQL và TMDB API đều được phân tích và làm rõ vai trò trong toàn bộ quy trình vận hành hệ thống. Đây là nền tảng quan trọng để chuyển sang chương tiếp theo, nơi sẽ đi sâu vào phân tích nghiệp vụ và thiết kế hệ thống chi tiết nhằm hiện thực hóa định hướng đã đề ra.

CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG.

2.1. Phân tích nghiệp vụ hệ thống

Hệ thống rạp chiếu phim được xây dựng nhằm phục vụ người dùng cuối (khách xem phim) với đầy đủ các chức năng từ đăng ký tài khoản đến đặt vé, xem lịch sử giao dịch và nhận gợi ý phim thông minh.

2.1.1. Vai trò chính: Người dùng (Customer)

- **Đăng ký tài khoản:**
  - Nhập thông tin cá nhân và nhận mã xác thực qua email (SMTP), mã được lưu trong Redis và có thời hạn 10 phút.
- **Đăng nhập:**
  - Sử dụng tài khoản đã xác thực để đăng nhập và nhận token JWT, phục vụ cho các request tiếp theo.
- **Xem thông tin phim:**
  - Bao gồm tiêu đề, mô tả, thể loại, đạo diễn, diễn viên, ảnh và trailer video.
  - Dữ liệu được gọi từ TMDB API, đảm bảo thông tin phong phú và được cập nhật liên tục.
- **Đặt vé xem phim:**
  - Chọn phim → chọn phòng chiếu → chọn suất chiếu → chọn ghế → xác nhận đặt vé.
  - Trước khi đặt, hệ thống kiểm tra trạng thái ghế qua API backend.
  - Nếu ghế đã được đặt trước đó nhưng phim đã đặt bị hủy (bởi người dùng), ghế sẽ tự động được mở lại để đặt.
- **Xem lịch sử đặt vé:**
  - Trong trang thông tin cá nhân, người dùng có thể xem các vé đã đặt, với trạng thái:
    - Đang hoạt động: vé vẫn còn hiệu lực.
    - Hết hiệu lực: suất chiếu đã diễn ra.
    - Đã hủy: vé đã bị hủy (trước thời điểm chiếu).
- **Tìm kiếm phim:**
  - Người dùng có thể tìm kiếm phim thông qua thanh tìm kiếm.

- **Nhận gợi ý phim:**

  - Dựa trên lịch sử đặt vé, hệ thống sử dụng mô hình AI LSTM để đề xuất danh sách phim phù hợp, gọi qua API Flask độc lập.

2.1.2. Vai trò của người quản trị viên:

- Xem thông tin ghế:
  - Xem thông tin các ghế được đặt.
- Xem thông tin phòng:
  - Xem thông tin phòng, đồng thời thêm, xóa phòng.
- Quản lý đặt vé:

- Xác nhận, từ chối yêu cầu đặt vé và yêu cầu hủy vé của khách hàng

- Quản lý thông tin người dùng:
  Xem thông tin người dùng đăng ký hệ thống.

2.2. Yêu cầu phi chức năng

2.2.1. Tính hiệu năng

- Hệ thống cần đảm bảo phản hồi nhanh, thời gian tải trang không vượt quá 3 giây đối với các chức năng cơ bản như đăng nhập, xem danh sách phim và đặt vé.
- API gợi ý phim AI phải trả về kết quả dưới 10 giây để không ảnh hưởng trải nghiệm người dùng.
- Thông tin phim được gọi từ TMDB API cần được cache lại để giảm độ trễ và hạn chế số lần truy vấn ra ngoài.

2.2.2. Tính bảo mật

- Người dùng phải đăng nhập (xác thực bằng JWT) trước khi thực hiện các chức năng như đặt vé, xem lịch sử giao dịch.
- Mật khẩu được mã hóa an toàn bằng thuật toán BCrypt trước khi lưu trữ vào hệ thống.
- Mã xác thực email khi đăng ký được gửi qua SMTP và lưu tạm thời trong Redis, có thời hạn 10 phút để chống lạm dụng.

2.2.3. Tính dễ sử dụng

- Giao diện được thiết kế bằng Nuxt.js (Vue 2) với bố cục trực quan, dễ sử dụng cho người dùng phổ thông.
- Các chức năng được bố trí logic, hỗ trợ người dùng thao tác đặt vé nhanh chóng chỉ trong vài bước.

2.2.4. Tính ổn định

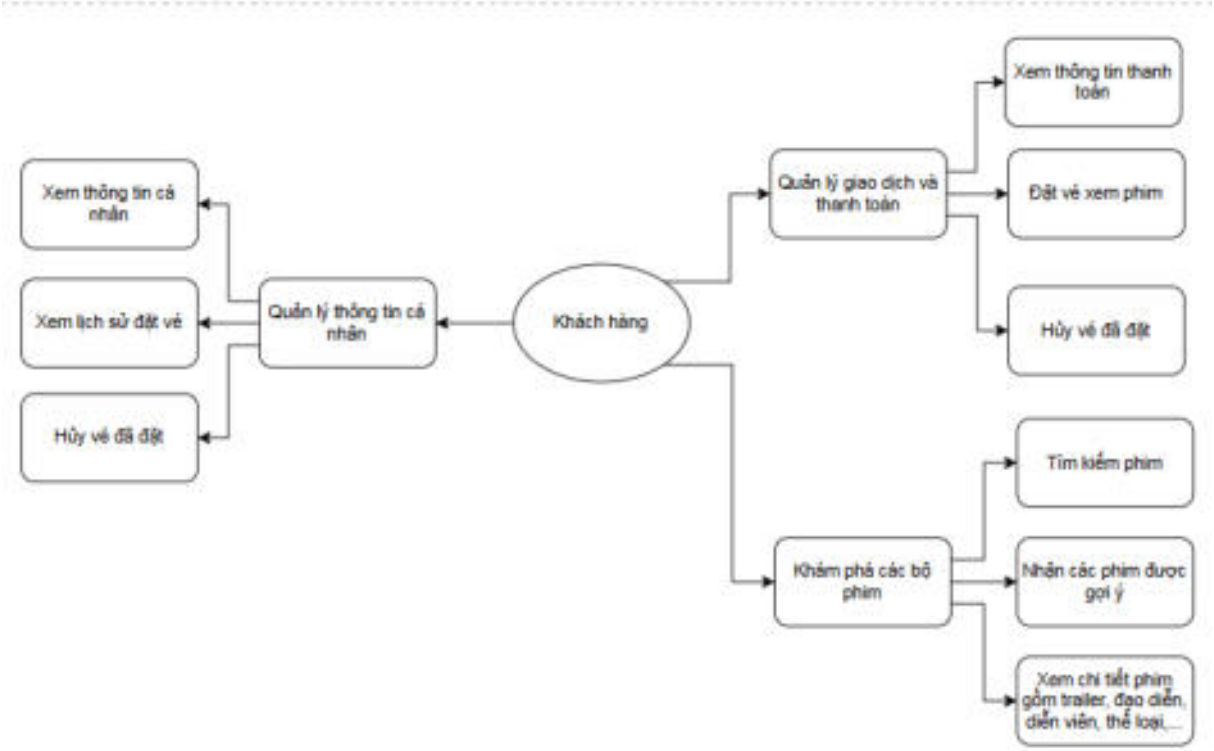
- Hệ thống được triển khai trên VPS với cấu hình ổn định, sử dụng Docker Compose để đảm bảo tất cả dịch vụ chạy đồng bộ.
- Có khả năng xử lý đồng thời nhiều truy vấn mà không ảnh hưởng đến các service khác (Flask, Redis, MySQL…).

2.2.5. Tính mở rộng

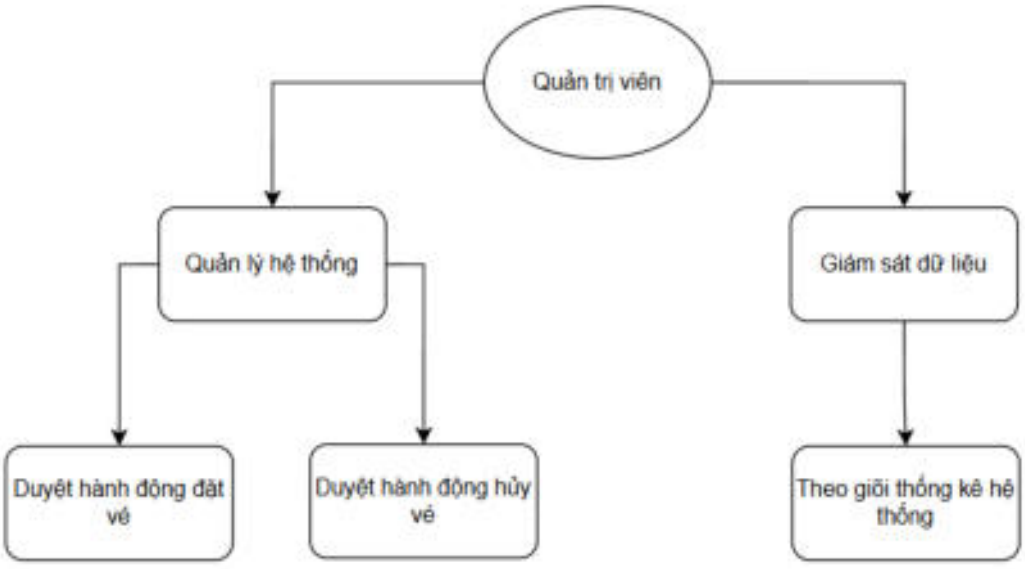
- Hệ thống được xây dựng theo kiến trúc microservices, có thể dễ dàng thêm mới các chức năng như thanh toán trực tuyến, chatbot trợ lý đặt vé, hoặc API dành cho mobile app.
- Cơ sở dữ liệu, mô hình AI và giao diện được tách rời nên có thể nâng cấp độc lập.

2.3. Thiết kế hệ thống

2.3.1. Sơ đồ phân rã chức năng

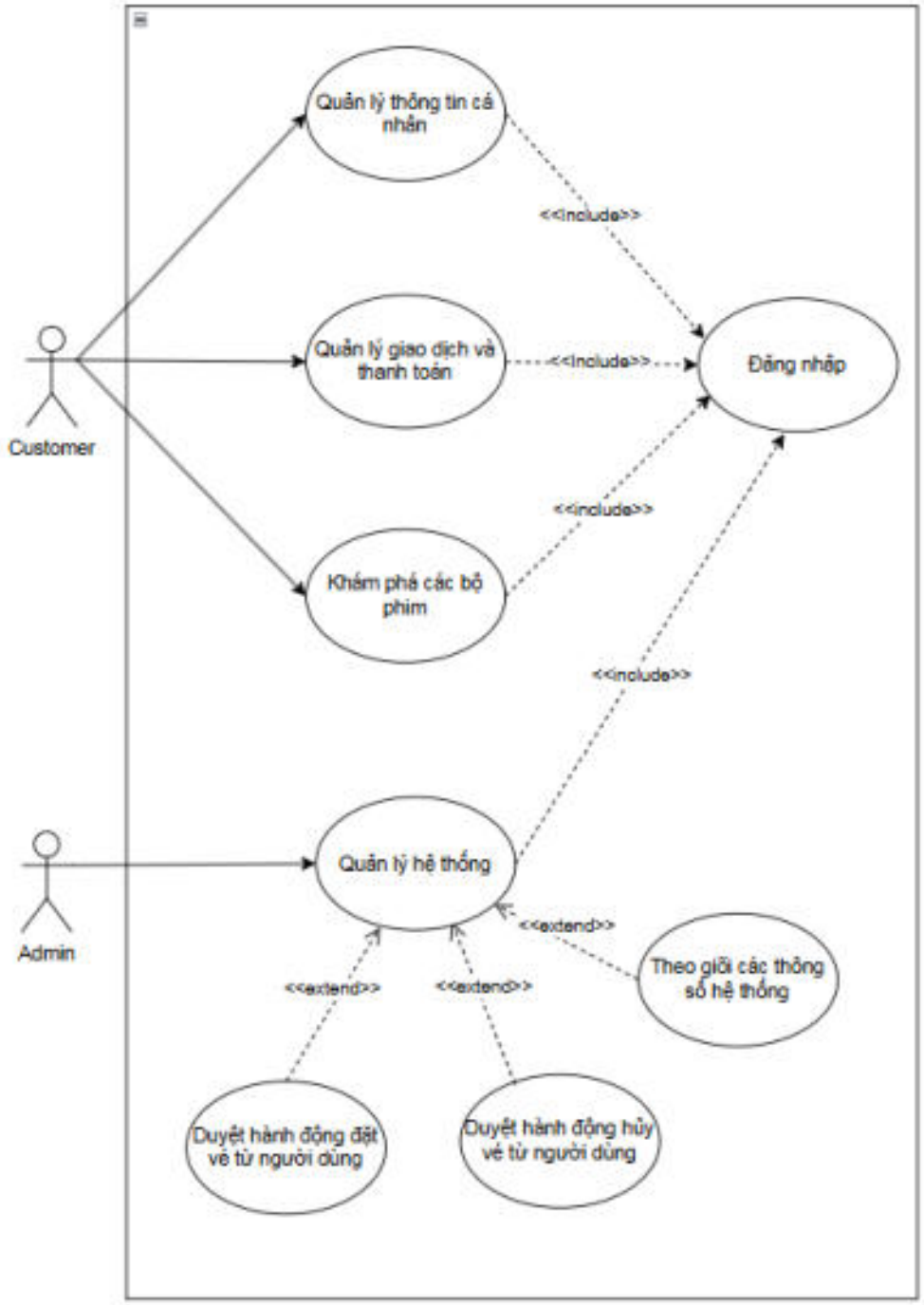


Hình 2.1: Sơ đồ phân rã chức năng Khách hàng.

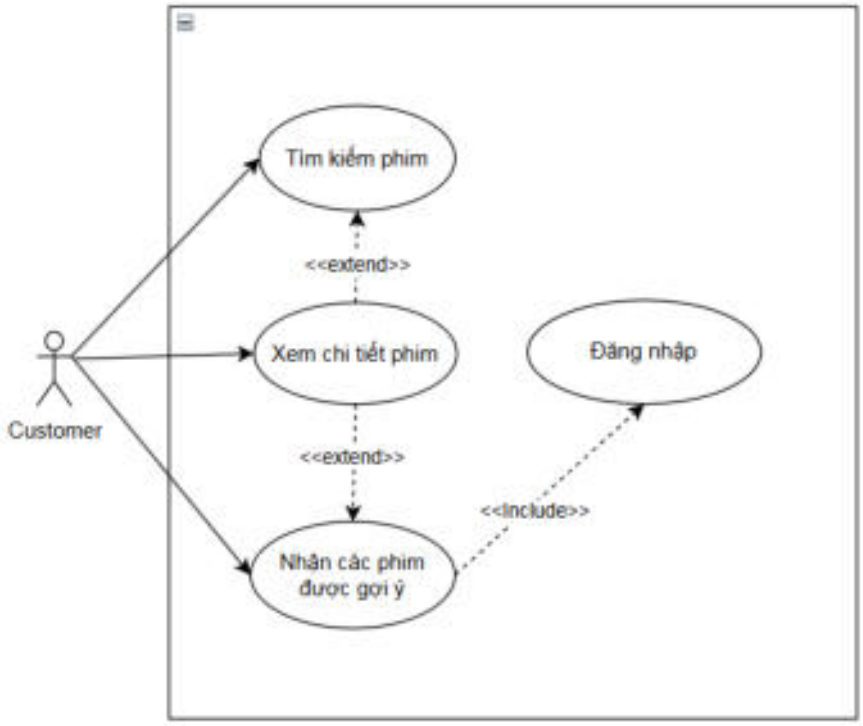


Hình 2.2: Sơ đồ phân rã chức năng quản trị viên.

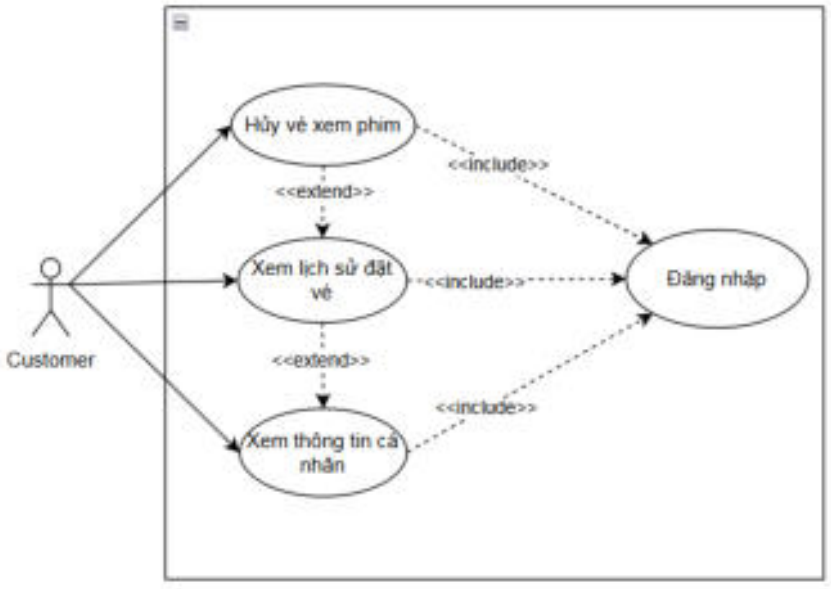
2.3.2. Sơ đồ ca sử dụng



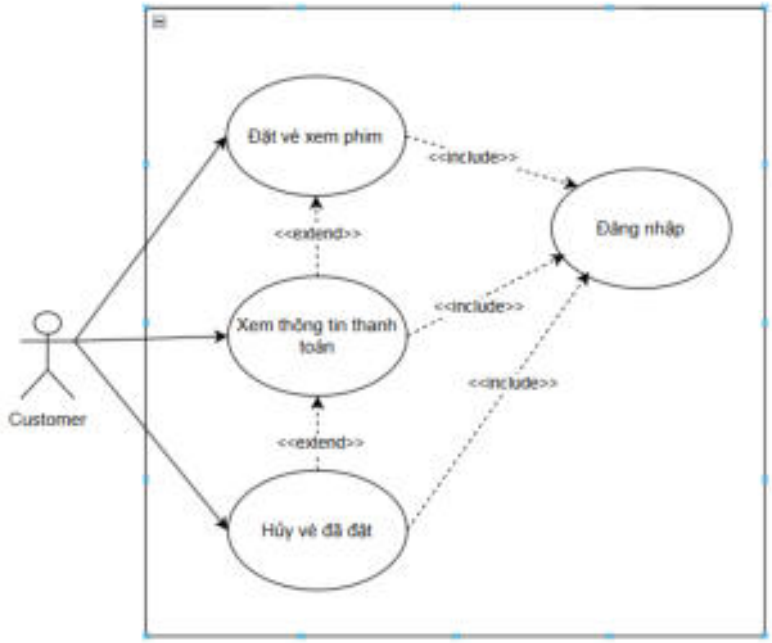
Hình 2.3: Sơ đồ ca sử dụng tổng quan.



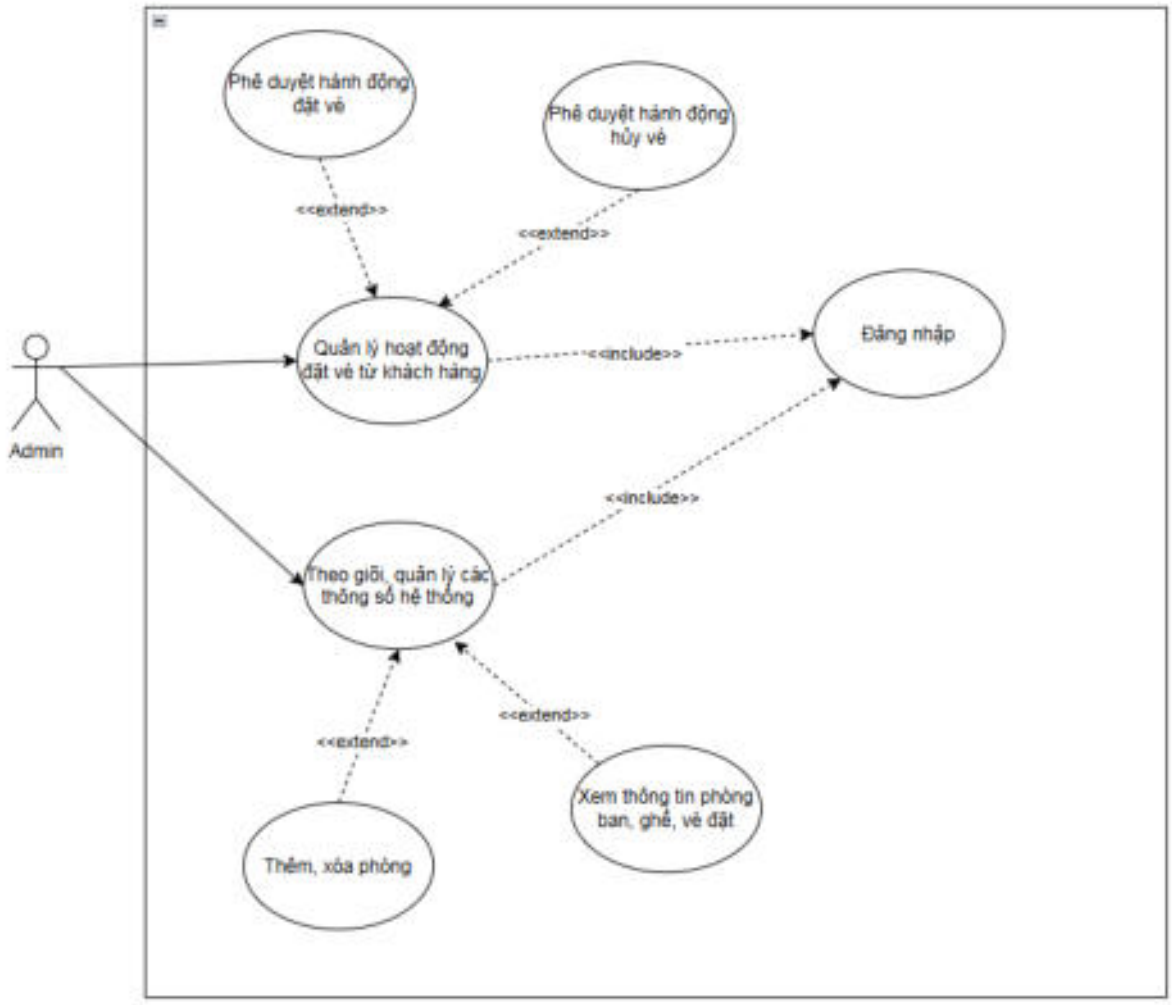
Hình 2.4: Sơ đồ ca sử dụng "Khám phá các bộ phim".



Hình 2.5: Sơ đồ ca sử dụng "Quản lý thông tin cá nhân".



Hình 2.6: Sơ đồ ca sử dụng "Quản lý giao dịch và thanh toán".



Hình 2.7: Sơ đồ ca sử dụng quản lý hệ thống.

2.3.3. Đặc tả ca sử dụng

- Ca sử dụng "Đăng nhập"

Bảng 2.1: Đặc tả ca sử dụng đăng nhập.

Tên ca sử dụng	**Đăng nhập**
Tác nhân	Khách hàng
Mô tả	Cho phép người dùng đăng nhập hệ thống
Điều kiện kích hoạt	Người dùng chưa đăng nhập vào hệ thống, truy cập vào đường link đăng nhập
Các bước thực hiện	1. Tại form đăng nhập, nhập tên tài khoản và mật khẩu.

	2. Nhấn nút login. 3. Hệ thống kiểm tra dữ liệu
Kết quả	Đăng nhập thành công, thanh thông báo hiển thị.
Trường hợp lỗi	Hệ thống kiểm tra người dùng không tồn tại hoặc nhập sai tài khoản hay mật khẩu.

- Ca sử dụng "Xem chi tiết phim":

Bảng 2.2: Đặc tả ca sử dụng "Xem chi tiết phim".

Tên ca sử dụng	**Xem chi tiết phim**
Tác nhân	Khách hàng
Mô tả	Cho phép khách hàng xem chi tiết phim như xem trailer phim, xem thể loại, đạo diễn, mô tả,...
Điều kiện kích hoạt	Khách hàng nhấn vào một bộ phim muốn xem chi tiết.
Các bước thực hiện	Tại màn hình danh sách phim, người dùng nhấn vào các thẻ chứa của mỗi bộ phim.
Kết quả	Người dùng được dẫn đến trang chi tiết phim.
Trường hợp lỗi	Đường dẫn phim bị lỗi.

- Ca sử dụng "Nhận phim được gợi ý"

Bảng 2.3: Đặc tả ca sử dụng "Nhận phim được gợi ý".

Tên ca sử dụng	**Nhận phim được gợi ý**
Tác nhân	Khách hàng
Mô tả	Khách hàng sẽ nhận được các phim gợi ý từ hệ thống dựa vào lịch sử đặt phim của khách hàng.
Điều kiện kích hoạt	Khách hàng đã đăng nhập thành công và nhấn vào đường dẫn tới trang gợi ý phim.
Các bước thực hiện	Truy cập vào mục "Gợi ý phim" ở thanh menu bên trái màn hình.
Kết quả	Trả về danh sách 10 bộ phim gợi ý.
Trường hợp lỗi	1. Người dùng chưa đặt bất kỳ vé nào từ khi đăng ký. 2. Lỗi không truy xuất được dữ liệu.

- Ca sử dụng “Tìm kiếm phim”:

Bảng 2.4: Đặc tả ca sử dụng "Tìm kiếm phim".

Tên ca sử dụng	**Tìm kiếm phim**
Tác nhân	Khách hàng.
Mô tả	Khách hàng có thể tìm kiếm các bộ phim thông qua thanh tìm kiếm.
Điều kiện kích hoạt	Khách hàng nhấn vào mục tìm kiếm ở thanh menu bên trái màn hình.
Các bước thực hiện	1. Nhấn vào mục tìm kiếm ở menu trái màn hình 2. Nhập từ khóa tìm kiếm.
Kết quả	Dữ liệu các phim có tên giống với từ khóa tìm kiếm sẽ được hiển thị lên màn hình chính.
Trường hợp lỗi	Lỗi truy xuất dữ liệu.

- Ca sử dụng “Xem thông tin cá nhân”

Bảng 2.5: Đặc tả ca sử dụng "Xem thông tin cá nhân".

Tên ca sử dụng	**Xem thông tin cá nhân**
Tác nhân	Khách hàng.
Mô tả	Khách hàng xem thông tin cá nhân của bản thân.
Điều kiện kích hoạt	Khách hàng đăng nhập, nhấn vào đường link dẫn đến trang.
Các bước thực hiện	Nhấn vào biểu tượng người dùng ở thanh menu và chọn phần profile.
Kết quả	Người dùng sẽ được dẫn đến trang xem thông tin cá nhân của họ.
Trường hợp lỗi	Chưa đăng nhập.

- Ca sử dụng “Xem lịch sử đặt vé”:

Bảng 2.6: Đặc tả ca sử dụng "Xem lịch sử đặt vé".

Tên ca sử dụng	**Xem lịch sử đặt vé**
Tác nhân	Khách hàng
Mô tả	Khách hàng có thể xem được lịch sử các vé đã đặt của mình, bao gồm vé đang còn hoạt động, vé đã hết hạn sử dụng và vé đã hủy.
Điều kiện kích hoạt	Khách hàng đã đăng nhập.
Các bước thực hiện	Nhấn vào biểu tượng người dùng ở thanh menu và chọn phần profile ở dropdown.
Kết quả	Khách hàng có thể xem được lịch sử đặt vé của mình.
Trường hợp lỗi	Lỗi truy xuất dữ liệu.

- Ca sử dụng “Hủy vé”:

Bảng 2.7: Đặc tả ca sử dụng "Hủy vé".

Tên ca sử dụng	**Hủy vé**
Tác nhân	Khách hàng, hệ thống.
Mô tả	Khách hàng có thể chủ động hủy vé hoặc hệ thống sẽ check xem vé đó đã quá hạn hay chưa để tự động hủy vé đó.
Điều kiện kích hoạt	Khách hàng đã đăng nhập/ Vé quá hạn.
Các bước thực hiện	1. Vào trang quản lý thông tin cá nhân của người dùng. 2. Nhấn vào nút hủy vé. 3. Xác nhận hủy.
Kết quả	Vé sẽ được hủy, trả lại dữ liệu ghế trống.
Trường hợp lỗi	Lỗi truy xuất dữ liệu.

- Ca sử dụng “Đặt vé xem phim”:

Bảng 2.8: Đặc tả ca sử dụng "Đặt vé xem phim".

Tên ca sử dụng	**Đặt vé xem phim**
Tác nhân	Khách hàng
Mô tả	Khách hàng có thể đặt vé xem phim mà khách hàng muốn xem.
Điều kiện kích hoạt	Khách hàng đã đăng nhập thành công.
Các bước thực hiện	1. Bấm vào nút “Ticket Now” để đến trang đặt vé. 2. Chọn phòng, chọn ngày chiếu, chọn xuất chiếu, chọn ghế. 3. Xác nhận đặt vé.
Kết quả	Vé sẽ được đặt thành công, ghi nhận vào database các ghế đã đặt.
Trường hợp lỗi	Khởi tạo dữ liệu bị lỗi.

- Ca sử dụng “Xem thông tin thanh toán”:

Bảng 2.9: Đặc tả ca sử dụng “Xem thông tin thanh toán”.

Tên ca sử dụng	**Xem thông tin thanh toán**
Tác nhân	Khách hàng
Mô tả	Khách hàng sẽ được xem thông tin thanh toán sau mỗi lần đặt vé xem phim thành công.
Điều kiện kích hoạt	Khách hàng đã đăng nhập vào hệ thống và đặt 1 vé xem phim thành công.
Các bước thực hiện	Hoàn thành đặt 1 vé xem phim thành công, màn hình sẽ tự động chuyển đến trang thông tin thanh toán.
Kết quả	Khách hàng sẽ xem được thông tin thanh toán, gồm tổng đơn, ghế, giờ chiếu, phòng.
Trường hợp lỗi	Thực thi quá lâu.

- Ca sử dụng Quản lý hoạt động đặt vé

Bảng 2.10: Đặc tả ca sử dụng quản lý hoạt động đặt vé.

Tên ca sử dụng	**Quản lý hoạt động đặt vé**
Tác nhân	Quản trị viên(Admin)
Mô tả	Quản trị viên quản lý phê duyệt các hành động đặt vé và hủy vé của khách hàng
Điều kiện kích hoạt	Admin đăng nhập vào trang quản lý
Các bước thực hiện	1. Admin vào trang quản lý vé 2. Admin thực hiện nhấn xác nhận hoặc từ chối đặt vé, hủy vé trong bảng danh sách.
Kết quả	Hệ thống sẽ cập nhật trạng thái vé và hiển thị lại trong danh sách.
Trường hợp lỗi	Thực thi quá lâu, lỗi truy xuất dữ liệu

- Ca sử dụng Theo giõi, quản lý các thông số hệ thống

Bảng 2.11: Đặc tả ca sử dụng theo giõi, quản lý thông số hệ thống.

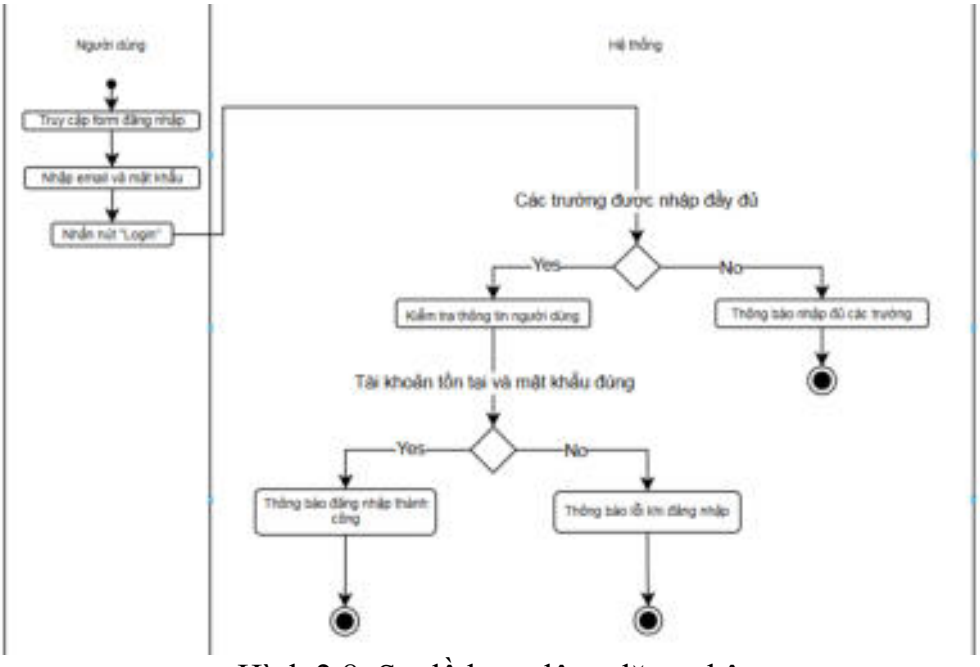
Tên ca sử dụng	**Theo giõi, quản lý các thông số hệ thống**
Tác nhân	Quản trị viên(Admin)
Mô tả	Quản trị viên theo giõi, quản lý các thông tin, dữ liệu của hệ thống rạp.
Điều kiện kích hoạt	Admin đăng nhập vào trang quản lý
Các bước thực hiện	1. Admin vào trang quản lý vé 2. Admin thực hiện xem các thông số trong các danh sách
Kết quả	Hệ thống sẽ hiển thị các thông tin tương ứng với các mục admin chọn
Trường hợp lỗi	Thực thi quá lâu, lỗi truy xuất dữ liệu

- Ca sử dụng thêm, xóa phòng

Tên ca sử dụng	**Thêm, xóa phòng**
Tác nhân	Quản trị viên(Admin)
Mô tả	Quản trị viên theo giõi, thêm hoặc xóa phòng
Điều kiện kích hoạt	Admin đăng nhập vào trang quản lý
Các bước thực hiện	1. Admin vào trang quản lý phòng 2. Admin thực hiện thêm hoặc xóa phòng
Kết quả	Hệ thống sẽ hiển thị các thông tin phòng tương ứng, các phòng đã được thêm hoặc xóa sẽ được cập nhật trong bảng
Trường hợp lỗi	Phòng có dữ liệu vé đã được đặt, lỗi truy xuất dữ liệu.

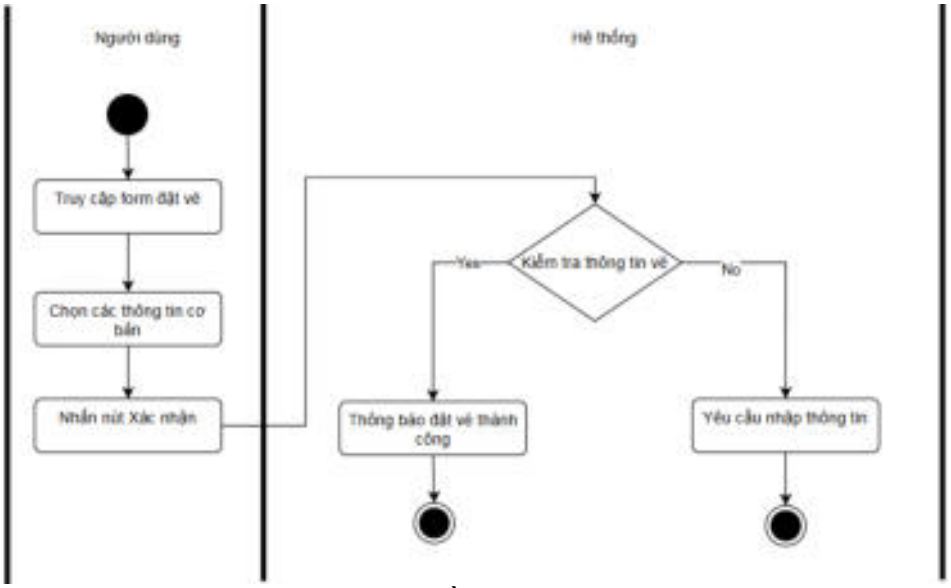
2.3.4. Sơ đồ hoạt động

- Sơ đồ hoạt động đăng nhập:



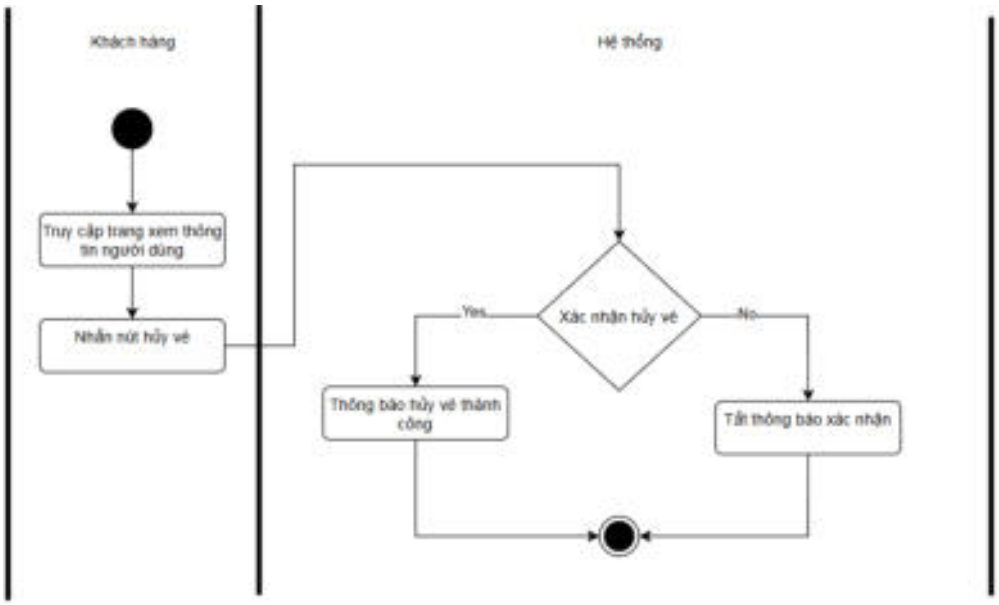
Hình 2.8: Sơ đồ hoạt động đăng nhập.

- Sơ đồ hoạt động đặt vé.



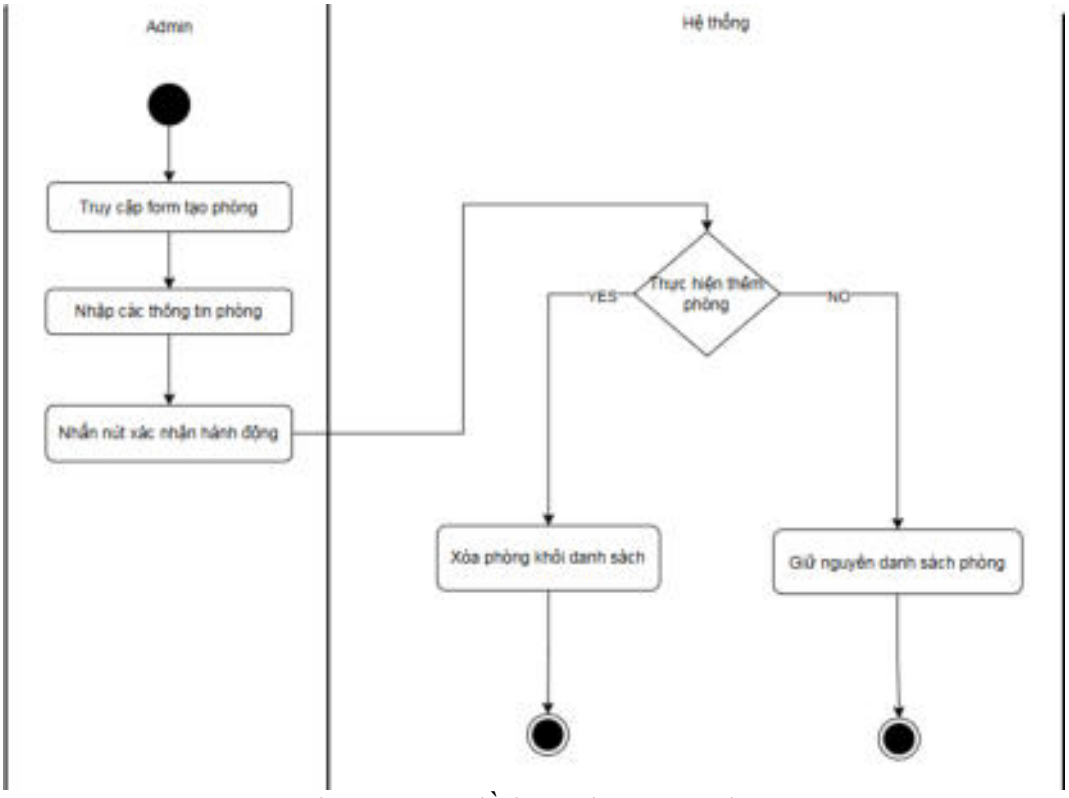
Hình 2.9: Sơ đồ hoạt động đặt vé.

- Sơ đồ hoạt động hủy vé.



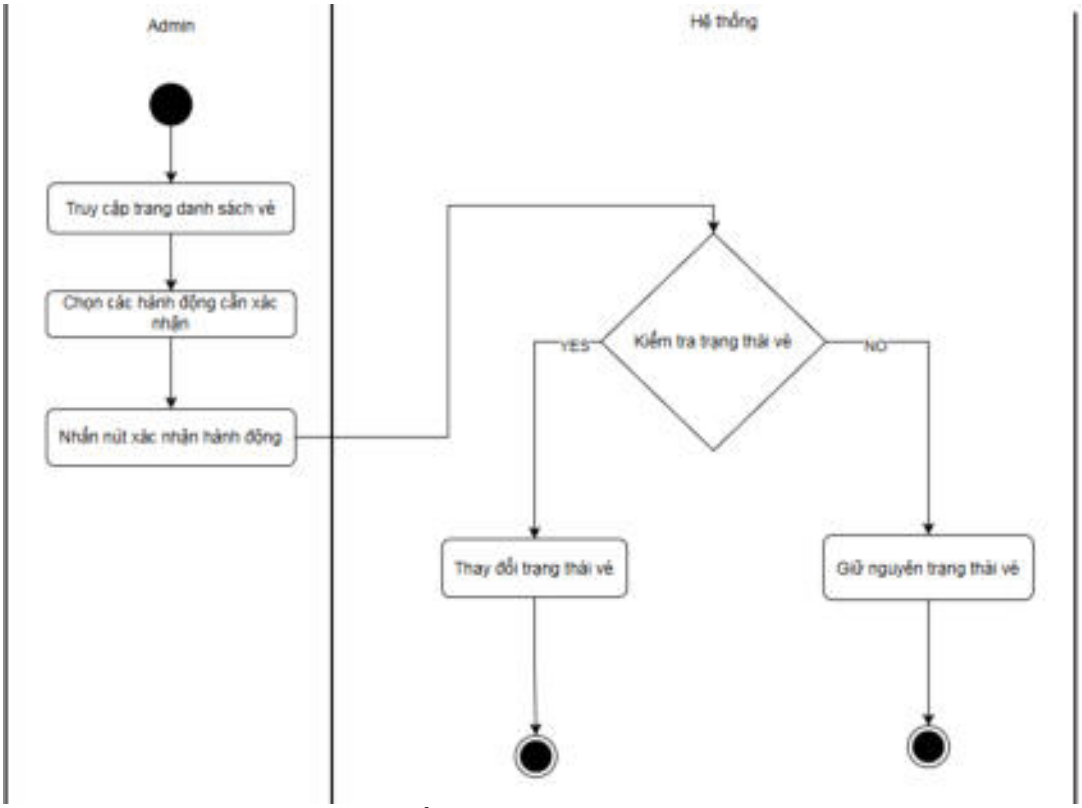
Hình 2.10: Sơ đồ hoạt động hủy vé.

- Sơ đồ hoạt động tạo phòng



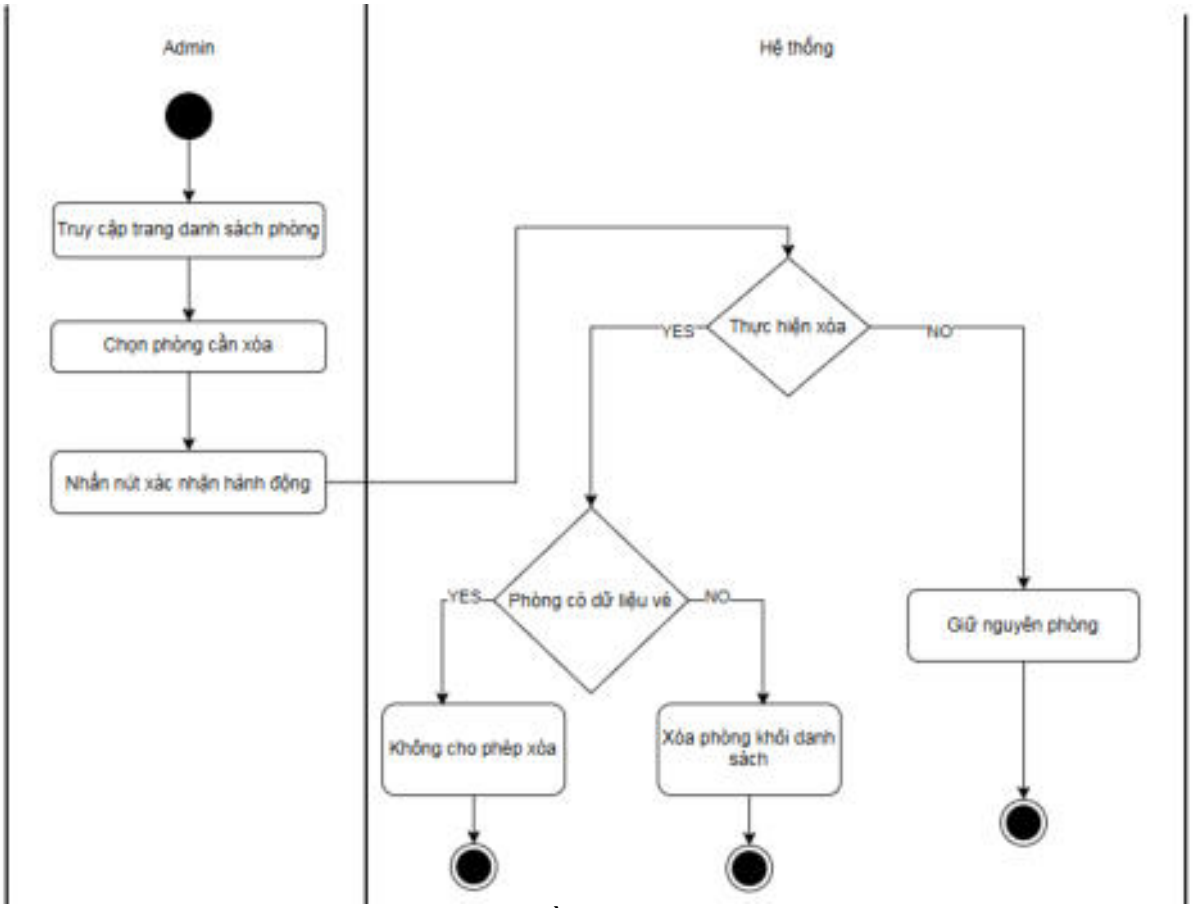
Hình 2.11: Sơ đồ hoạt động tạo phòng.

- Sơ đồ hoạt động quản lý trạng thái vé



Hình 2.12: Sơ đồ hoạt động quản lý trạng thái vé.

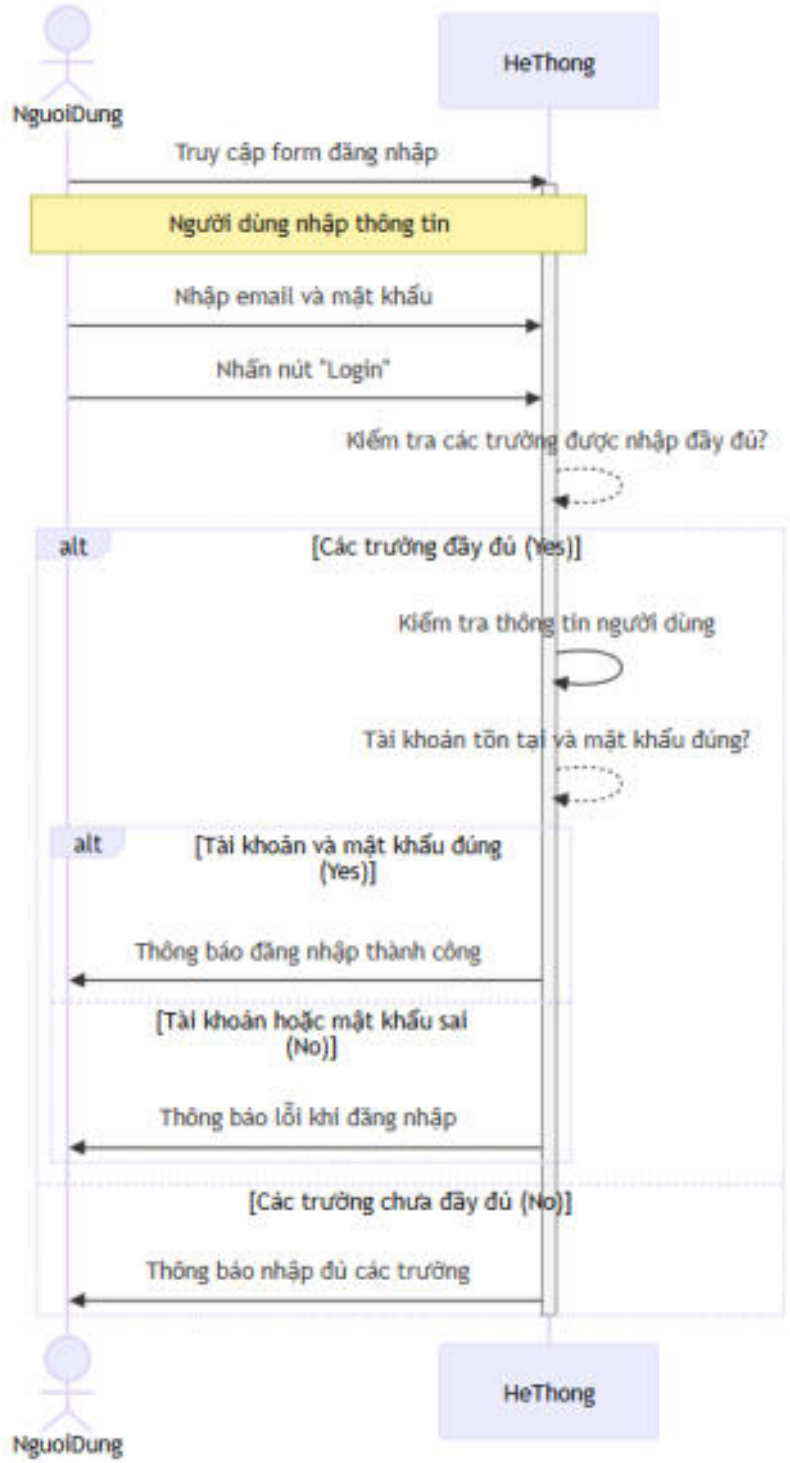
- Sơ đồ hoạt động xóa phòng



Hình 2.13: Sơ đồ hoạt động xóa phòng.

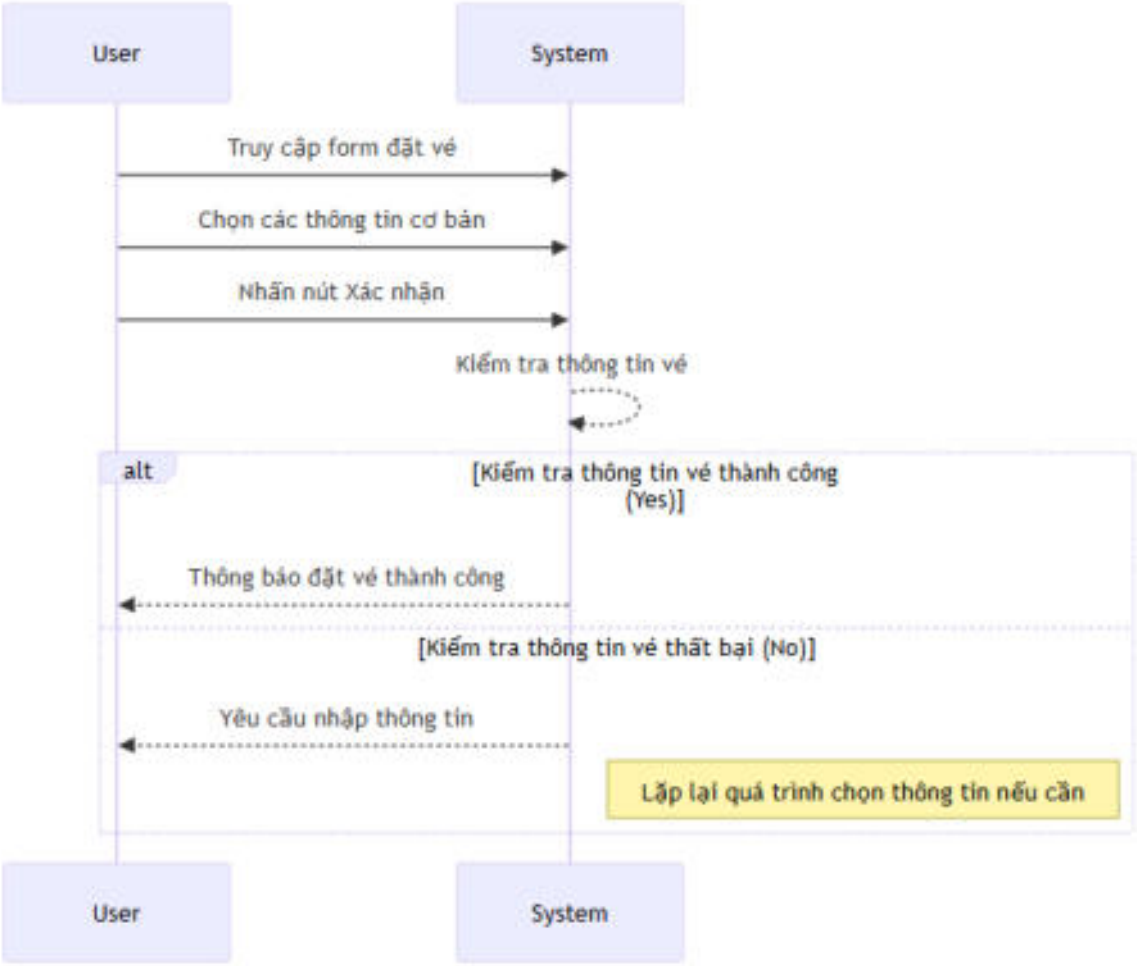
2.3.5. Sơ đồ tuần tự

- Sơ đồ tuần tự "đăng nhập":



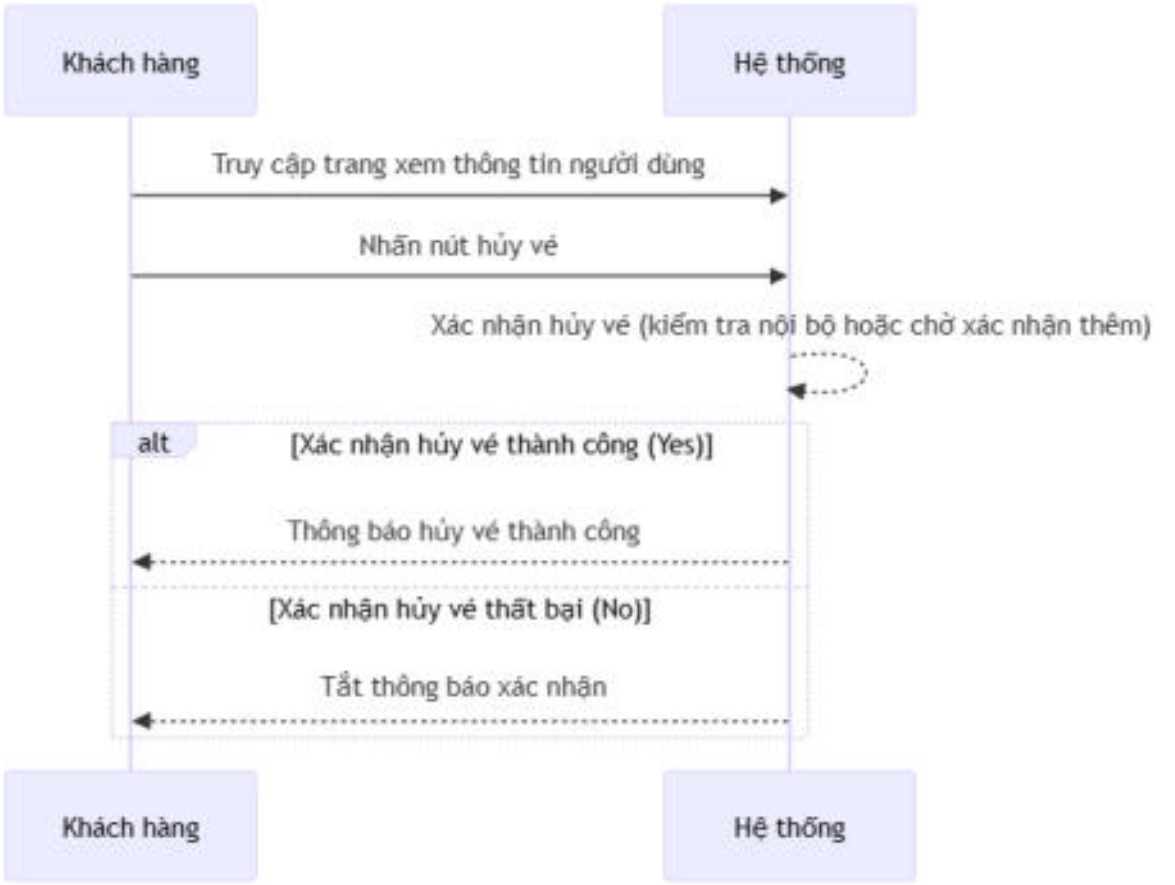
Hình 2.14: Sơ đồ tuần tự đăng nhập.

- Sơ đồ tuần tự đặt vé:



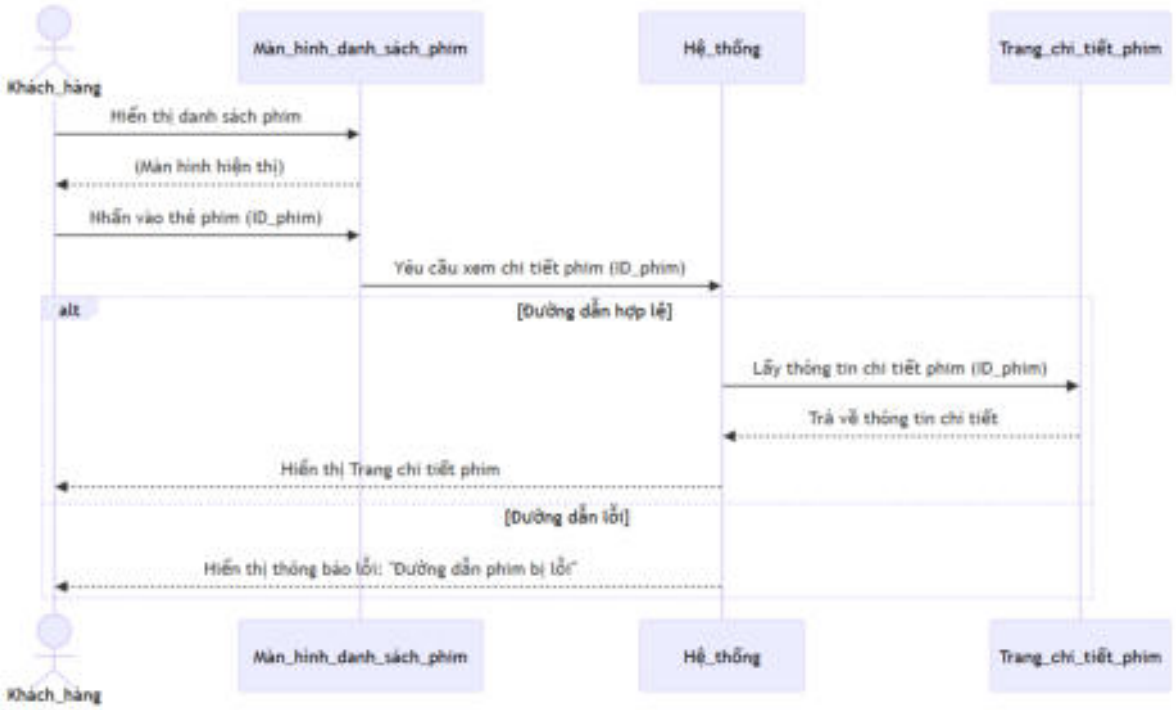
Hình 2.15: Sơ đồ tuần tự đặt vé.

- Sơ đồ tuần tự hủy vé:



Hình 2.16: Sơ đồ tuần tự chức năng hủy vé.

- Sơ đồ tuần tự xem chi tiết phim:



Hình 2.17: Sơ đồ tuần tự xem chi tiết phim.

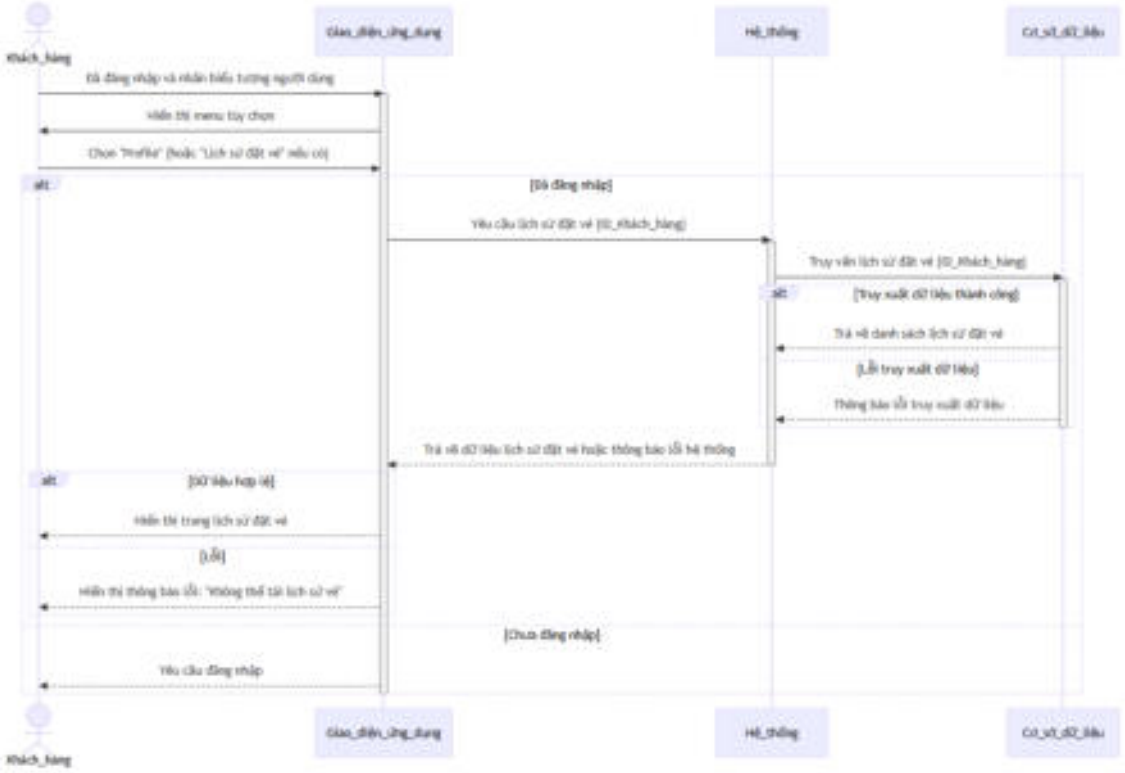
- Sơ đồ tuần tự Nhận phim được gợi ý:

Hình 2.18: Sơ đồ tuần tự nhận phim được gợi ý.

- Sơ đồ tuần tự Xem thông tin cá nhân:

Hình 2.19: Sơ đồ tuần tự xem thông tin cá nhân.

- Sơ đồ tuần tự xem lịch sử đặt vé:

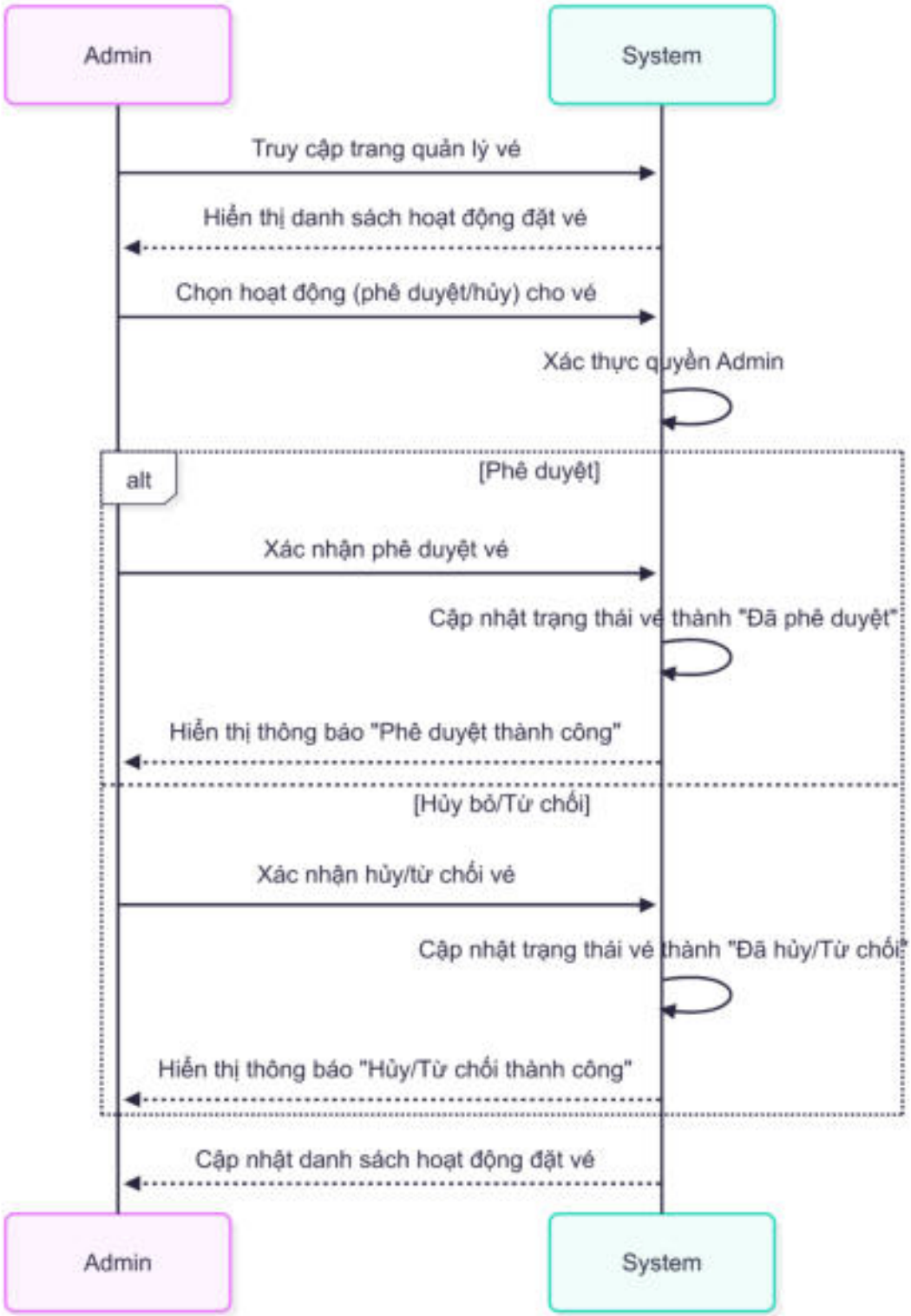


Hình 2.20: Sơ đồ tuần tự xem lịch sử đặt vé.

- Sơ đồ tuần tự xem thông tin thanh toán:

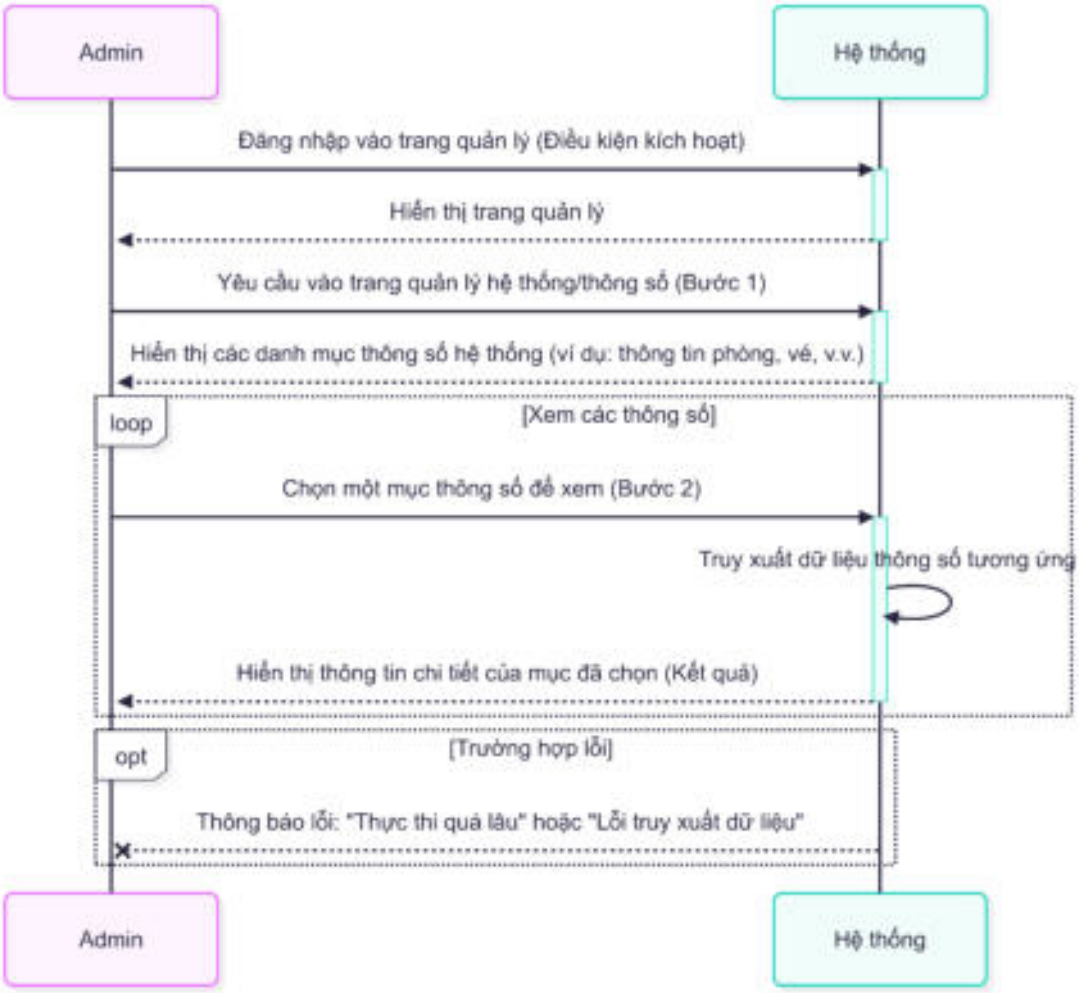
Hình 2.21: Sơ đồ tuần tự xem thông tin thanh toán.

- Sơ đồ tuần tự quản lý hành động đặt vé



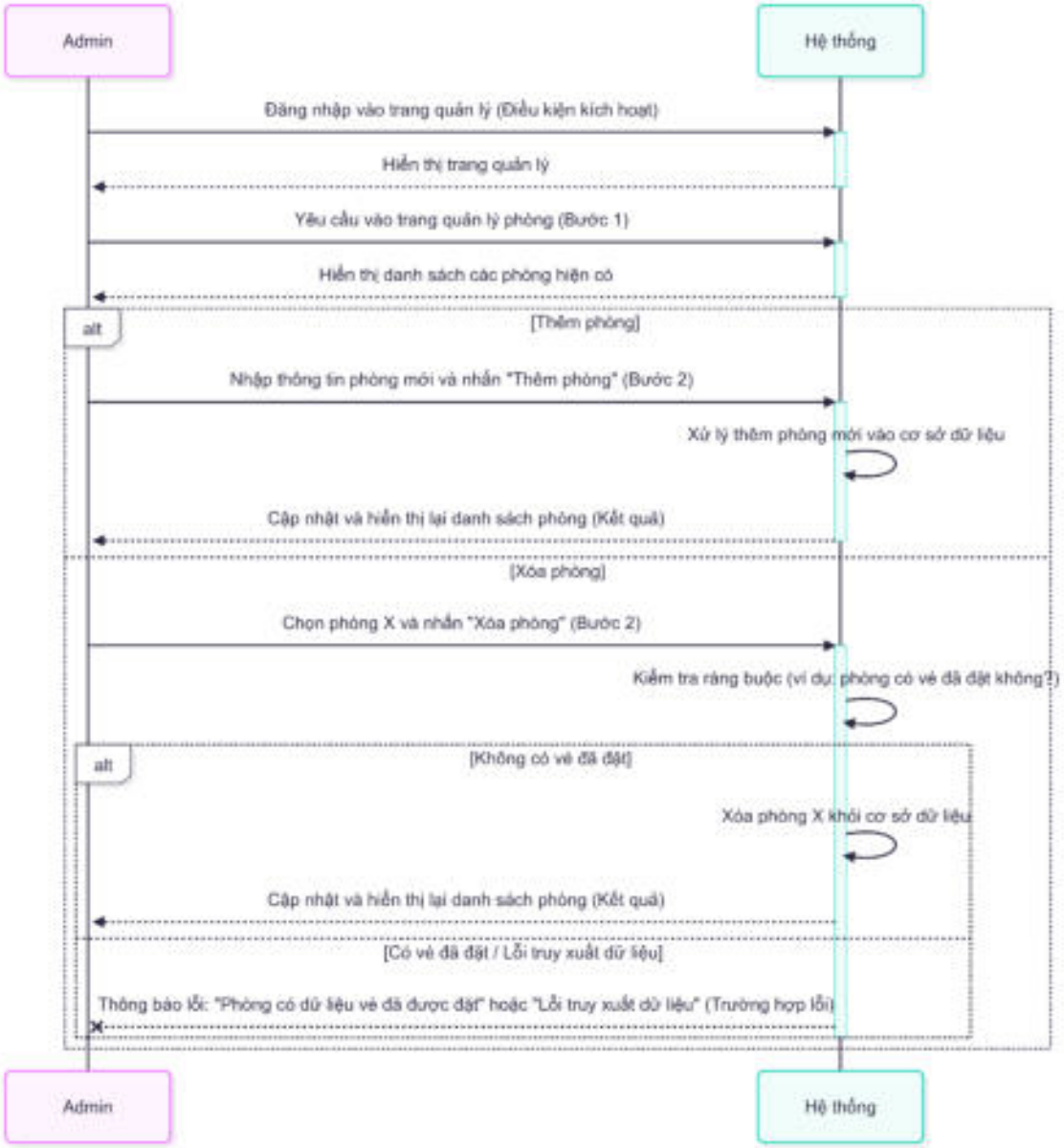
Hình 2.22: Sơ đồ tuần tự hành động đặt vé.

- Sơ đồ tuân tự Theo giõi, quản lý thông số hệ thống.



Hình 2.23: Sơ đồ tuần tự theo giõi, quản lý thông số hệ thống.

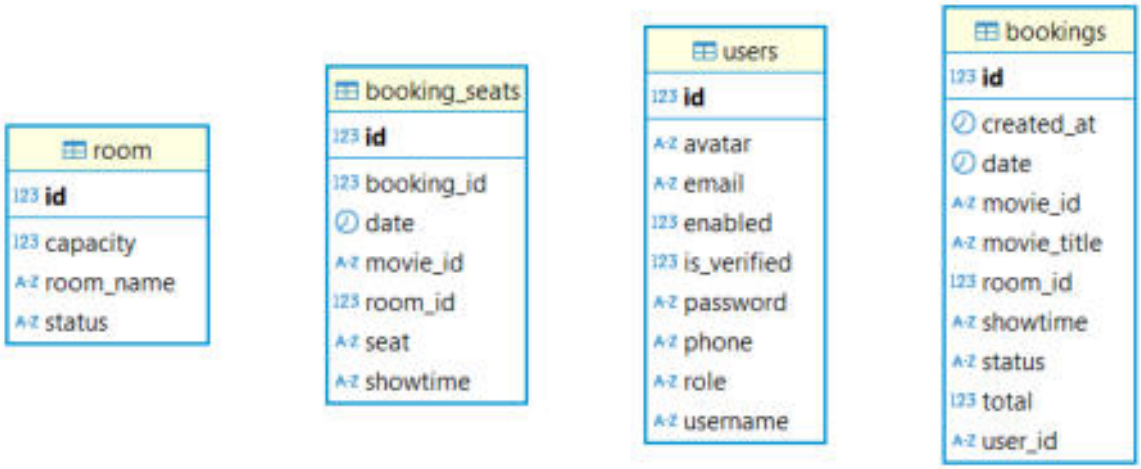
- Sơ đồ tuần tự thêm, xóa phòng(admin)



Hình 2.24: Sơ đồ tuần tự thêm, xóa phòng.

2.3.6. Thiết kế cơ sở dữ liệu

2.3.6.1. Module quản lý dữ liệu & thiết kế cơ sở dữ liệu



Hình 2.25: Sơ đồ cơ sở dữ liệu hệ thống.

- Cấu trúc bảng room:

Bảng 2.12: Cấu trúc bảng room.

Tên cột	Kiểu dữ liệu	Mô tả	Ghi chú
id	bigint	Mã phòng	Khóa chính, tự tăng
capacity	int	Sức chứa tối đa của phòng	Có thể null
room_name	varchar(255)	Tên phòng chiếu	Có thể null
status	varchar(255)	Trạng thái phòng	Có thể null

- Cấu trúc bảng User:

Bảng 2.13: Cấu trúc bảng user.

Tên cột	**Kiểu dữ liệu**	**Mô tả**	**Ghi chú**
id	bigint	Mã người dùng	Khóa chính, tự tăng
avatar	varchar(255)	Link ảnh đại diện (nếu có)	Có thể null
email	varchar(255)	Email người dùng	Có thể null
enabled	bit(1)	Trạng thái hoạt động	0: vô hiệu, 1: hiệu
is_verified	bit(1)	Đã xác thực email hay chưa	0 hoặc 1
password	varchar(255)	Mật khẩu (đã mã hóa)	Có thể null
phone	varchar(255)	Số điện thoại	Có thể null
role	varchar(255)	Vai trò người dùng (USER/ADMIN)	Có thể null
username	varchar(255)	Tên đăng nhập	Có thể null

- Cấu trúc bảng bookings:

Bảng 2.14: Cấu trúc bảng bookings.

Tên cột	**Kiểu dữ liệu**	**Mô tả**	**Ghi chú**
id	bigint	Mã đơn đặt vé	Khóa chính, tự tăng
created_at	datetime(6)	Thời điểm tạo	Có thể null
date	date	Ngày chiếu phim	Không null
movie_id	varchar(255)	Mã phim	Có thể null
movie_title	varchar(255)	Tên phim	Có thể null
room_id	bigint	Mã phòng chiếu	Có thể null
showtime	varchar(255)	Suất chiếu	Có thể null
status	varchar(255)	Trạng thái đặt vé (WAITING, SUCCESS, WAITTING_ACTIVE, WAITING_CANCEL, EXPIRED, CANCELED)	Có thể null
total	double	Tổng số tiền	Có thể null
user_id	varchar(255)	Mã người dùng	Có thể null

- Cấu trúc bảng booking_seats:

Bảng 2.15: Cấu trúc bảng booking_seats.

Tên cột	**Kiểu dữ liệu**	**Mô tả**	**Ghi chú**
id	bigint	Mã định danh	Khóa chính, tự tăng
booking_id	bigint	Mã đặt vé liên kết	Khóa ngoại (đề xuất)
date	date	Ngày chiếu phim	Không null
movie_id	varchar(255)	Mã phim	Có thể null
room_id	bigint	Mã phòng chiếu	Không null
seat	varchar(255)	Ghế được đặt	Có thể null
showtime	varchar(255)	Suất chiếu	Có thể null

2.4. Kết luận chương

Chương này đã trình bày chi tiết quá trình phân tích và thiết kế hệ thống cho dự án **"Hệ thống rạp chiếu phim tích hợp AI gợi ý phim"**. Bắt đầu từ việc phân tích nghiệp vụ người dùng, chương đã xác định rõ các chức năng chính như đăng ký, đăng nhập, xem thông tin phim, đặt vé, và nhận gợi ý phim dựa trên lịch sử.

Tiếp theo, các sơ đồ phân tích như **sơ đồ ca sử dụng**, **sơ đồ phân rã chức năng**, **sơ đồ tuần tự** và **sơ đồ kiến trúc tổng quan** đã giúp mô tả trực quan quy trình xử lý và luồng tương tác giữa các thành phần của hệ thống.

Đặc biệt, phần thiết kế cơ sở dữ liệu đã chỉ ra các bảng chính như room, users, bookings, booking_seats cùng cấu trúc chi tiết, giúp đảm bảo lưu trữ dữ liệu logic và thuận tiện cho việc mở rộng sau này. Ngoài ra, phương pháp AI dựa trên mạng **LSTM kết hợp BERT embedding** cũng được lựa chọn để hỗ trợ tính năng gợi ý phim một cách chính xác và cá nhân hóa.

Những thiết kế được đề xuất trong chương này là cơ sở quan trọng để triển khai hệ thống trong chương tiếp theo. Đồng thời, quá trình phân tích nghiệp vụ cũng giúp xác định rõ các yêu cầu chức năng và phi chức năng, đảm bảo tính khả thi và hiệu quả cho toàn bộ hệ thống.

CHƯƠNG 3: TRIỂN KHAI VÀ KIỂM THỬ HỆ THỐNG.

3.1 Công cụ phát triển

Trong quá trình xây dựng hệ thống **rạp chiếu phim tích hợp AI gợi ý phim**, em đã sử dụng nhiều công cụ và nền tảng khác nhau để phục vụ việc lập trình, huấn luyện mô hình, quản lý cơ sở dữ liệu, kiểm thử và triển khai. Dưới đây là các công cụ chính được sử dụng

3.1.1. Ngôn ngữ và nền tảng lập trình

Bảng 3.1: Ngôn ngữ và nền tảng lập trình.

Công nghệ	Vai trò
Java 11	Xây dựng backend với Spring Boot (RESTful API, xử lý bảo mật)
Python 3.x	Huấn luyện mô hình AI gợi ý phim bằng LSTM + BERT
Vue 2 / Nuxt.js	Xây dựng frontend SPA thân thiện, dễ bảo trì

3.1.2. IDE và môi trường lập trình

Bảng 3.2: IDE và môi trường lập trình.

Công cụ	Mục đích sử dụng
Visual Studio Code	Phát triển backend Java Spring Boot
Visual Studio Code	Lập trình frontend Vue/Nuxt.js và Flask API
Google Colab	Huấn luyện mô hình LSTM trên GPU

3.1.3. Quản lý mã nguồn và cộng tác

Bảng 3.3: Quản lý mã nguồn và cộng tác.

Công cụ	Vai trò
Git + GitHub	Lưu trữ mã nguồn, version control, deploy thủ công

3.1.4. Cơ sở dữ liệu

Bảng 3.4: Cơ sở dữ liệu.

Công cụ	Vai trò
MySQL	Lưu trữ thông tin người dùng, vé đặt, phim
Redis	Cache mã xác thực email tạm thời

3.1.5. Quản lý container và triển khai

Bảng 3.5: Quản lý container và triển khai.

Công cụ	Vai trò
Docker	Đóng gói backend, frontend và Flask server thành container riêng biệt
Docker Compose	Điều phối các container trong môi trường tích hợp

3.1.6. Gửi mã xác thực

Bảng 3.6: Gửi mã xác thực.

Công cụ	Vai trò
Gmail SMTP	Gửi mã xác nhận đến email người dùng khi đăng ký

3.1.7. Công cụ hỗ trợ kiểm thử

Bảng 3.7: Công cụ hỗ trợ kiểm thử.

Công cụ	Vai trò
Postman	Gửi request để test REST API backend
DBeaver	Kiểm tra trực quan dữ liệu trong MySQL

3.2. Triển khai quá trình gợi ý phim

Hệ thống gợi ý phim trong dự án được triển khai dựa trên mô hình học sâu **LSTM (Long Short-Term Memory)** kết hợp với **BERT (Bidirectional Encoder Representations from Transformers)** để trích xuất đặc trưng ngữ nghĩa từ mô tả phim. Mục tiêu là gợi ý các phim phù hợp dựa trên lịch sử đặt vé của người dùng.

3.2.1. Dữ liệu đầu vào

Dữ liệu được lấy từ tập tin TMDB_movie_dataset_v11.csv, bao gồm:

- id: ID của mỗi bộ phim
- overview: mô tả phim (tiếng Anh)
- genres: thể loại
- vote_average, popularity: điểm đánh giá và mức độ phổ biến

Sau khi tải và xử lý, dữ liệu được lọc để giữ lại **top 10.000 phim phổ biến nhất**.

3.2.2. Tiền xử lý dữ liệu

- Làm sạch mô tả (loại bỏ ký tự đặc biệt, bỏ phim không rõ nội dung)
- Mã hóa thể loại bằng MultiLabelBinarizer
- Chuẩn hóa vote_average và popularity
- Dùng **BERT** để tạo embedding cho overview
- Kết hợp các đặc trưng thành vector đầu vào cho mỗi phim

3.2.3. Tạo dữ liệu huấn luyện

- Với mỗi phim, tìm **3 phim tương tự nhất** dựa trên cosine similarity
- Tạo chuỗi X gồm 3 phim đầu vào, y là phim đích cần dự đoán
- Tập dữ liệu có hàng ngàn mẫu đầu vào ở dạng (X, y)

3.2.4. Kiến trúc mô hình

- Mô hình gồm 2 tầng LSTM và 1 Dense layer đầu ra
- Sử dụng hàm mất mát MeanSquaredError, tối ưu hóa bằng Adam
- Đánh giá độ tương đồng bằng CosineSimilarity

3.2.5. Huấn luyện mô hình

3.2.5.1. Chi tiết các bước huấn luyện

- **Thu thập và Tiền xử lý dữ liệu:**
  - Dữ liệu phim (từ file CSV TMDB_movie_dataset_v11.csv) được tải và làm sạch để loại bỏ các phim thiếu thông tin quan trọng (id, overview, genres), các mô tả không hợp lệ, và các thể loại không nằm trong danh sách định trước (ALL_GENRES).
  - Chỉ giữ lại 10.000 phim có popularity cao nhất sau khi sắp xếp, đây là một cách để tập trung vào các phim phổ biến và có thể đại diện tốt hơn.
- **Mã hóa Thể loại (Genre Encoding):**
  - Sử dụng MultiLabelBinarizer để chuyển đổi các thể loại của mỗi phim thành một **vector nhị phân (one-hot encoding)**. Ví dụ, nếu một phim có thể loại "Action, Drama", nó sẽ được biểu diễn bằng một vector mà tại vị trí của "Action" và "Drama" là 1, các vị trí khác là 0. Điều này giúp máy tính hiểu được các thể loại của phim.
- **Tạo Embedding cho Mô tả (Overview Embeddings) bằng BERT:**

  - Sử dụng mô hình BERT (bert-base-uncased), một mô hình ngôn ngữ lớn, để chuyển đổi overview (tóm tắt nội dung) của mỗi phim thành một **vector số (embedding)**. Embedding này là một biểu diễn dày đặc (dense representation) chứa đựng ý nghĩa ngữ nghĩa của mô tả. Các phim có nội dung tương tự sẽ có các vector embedding gần nhau trong không gian vector.
- **Kết hợp các Thuộc tính thành Feature Vector:**
  - Đây là một bước cực kỳ quan trọng. Đối với mỗi phim, các thuộc tính khác nhau được kết hợp thành một **feature vector (vector đặc trưng)** duy nhất. Vector này bao gồm:
    - **Vector thể loại:** Từ MultiLabelBinarizer (ví dụ: 19 chiều).
    - **Vector embedding mô tả:** Từ BERT (768 chiều).
    - **Điểm đánh giá trung bình (Vote Average):** Chuẩn hóa về khoảng [0, 1] (1 chiều).
    - **Độ phổ biến (Popularity):** Chuẩn hóa về một khoảng hợp lý (1 chiều).
  - Các vector này được **nối lại (concatenate)** với nhau để tạo thành một vector dài hơn, đại diện toàn diện cho một bộ phim. Đây chính là "đầu vào" mà mô hình LSTM sẽ học.
- **Tạo Ma trận Tương đồng (Similarity Matrix):**
  - Sử dụng độ **tương đồng cosine (cosine similarity)** giữa các feature vector của tất cả các phim để tạo ra một ma trận. Ma trận này cho biết mức độ tương đồng giữa bất kỳ cặp phim nào trong tập dữ liệu.
- **Tạo chuỗi dữ liệu cho LSTM:**
  - Đây là phần sử dụng "lịch sử xem". Với mỗi bộ phim trong tập dữ liệu, mô hình tìm **sequence_length (ví dụ: 3) bộ phim tương đồng nhất** với nó (dựa trên ma trận tương đồng đã tính toán).
  - Các feature vector của sequence_length phim tương đồng này được ghép lại thành một **chuỗi đầu vào (input sequence)** cho LSTM (X).
  - Feature vector của **chính bộ phim hiện tại** (y) được sử dụng làm **đầu ra mục tiêu (target output)** mà LSTM cần dự đoán.
  - **Ý tưởng:** Nếu người dùng đã xem 3 bộ phim tương tự X, Y, Z, thì bộ phim hiện tại (là X, Y hoặc Z) có thể được coi là bộ phim "tiếp theo" trong chuỗi quan tâm của họ. Mô hình sẽ học cách từ các feature của X, Y, Z để dự đoán feature của X (hoặc một phim tương tự).
- **Xây dựng và Biên dịch Mô hình LSTM:**
  - Mô hình được xây dựng với các tầng LSTM, nhận vào một chuỗi các feature vector và trả về một feature vector duy nhất (dự đoán). Các tầng Dropout được thêm vào để chống quá khớp (overfitting).
  - Mô hình được biên dịch với hàm loss là **Mean Squared Error (MSE)**, vì nó đang cố gắng dự đoán một vector số (bài toán hồi quy). Adam là bộ tối ưu hóa và CosineSimilarity được dùng làm metric đánh giá hiệu suất.
- **Huấn luyện Mô hình:**
  - Mô hình được huấn luyện trên dữ liệu X và y. Các callbacks như EarlyStopping (dừng sớm nếu không cải thiện) và ReduceLROnPlateau

(giảm tốc độ học nếu mất mát không giảm) được sử dụng để tối ưu quá trình huấn luyện.
- Thực hiện trên Google Colab
- Tập huấn luyện chia tỷ lệ 80:20
- Kết quả sau huấn luyện đạt:
  - **Cosine Similarity trên tập validation ~ 0.945**
  - **Loss ~ 0.0350**

3.2.6. Tại sao phim này được gợi ý?

Giả sử bạn đã xem 3 bộ phim gần đây: **Phim A, Phim B, Phim C**.

1. **Lấy Feature Vector:** Hệ thống sẽ lấy các feature vector của Phim A, Phim B, Phim C.
   - VA=combine(genresA,embeddingA,voteA,popA)
   - VB=combine(genresB,embeddingB,voteB,popB)
   - VC=combine(genresC,embeddingC,voteC,popC)
2. **Đưa vào LSTM:** Chuỗi [V_A, V_B, V_C] sẽ được đưa vào mô hình LSTM đã được huấn luyện.
3. **LSTM dự đoán Feature Vector:** Mô hình LSTM sẽ học các mối quan hệ và xu hướng trong chuỗi này và dự đoán một **vector đặc trưng tiềm năng** (V_{pred}) cho bộ phim tiếp theo mà bạn có thể thích. V_{pred} này đại diện cho "kiểu phim" mà bạn đang tìm kiếm dựa trên lịch sử xem.
4. **Tìm phim tương đồng nhất:** Hệ thống sau đó sẽ tính toán độ **tương đồng cosine** giữa V_{pred} này với feature vector của **tất cả các phim còn lại** trong cơ sở dữ liệu (những phim bạn chưa xem).
5. **Gợi ý phim:** Các bộ phim có độ tương đồng cosine cao nhất với V_{pred} sẽ là những bộ phim được gợi ý.

Ví dụ cụ thể:

Nếu bạn xem Phim A (Hành động, Khoa học viễn tưởng, Kỹ xảo đẹp), Phim B (Hành động, Giật gân, Có yếu tố mật vụ), và Phim C (Khoa học viễn tưởng, Du hành thời gian, Hiệu ứng hình ảnh đỉnh cao):

- LSTM sẽ học được rằng bạn có xu hướng xem các phim **Hành động** và **Khoa học viễn tưởng**, đặc biệt là những phim có **kỹ xảo/hiệu ứng hình ảnh mạnh mẽ** và có thể có yếu tố **mật vụ/giật gân**.
- Khi bạn đưa V_A, V_B, V_C vào LSTM, nó sẽ dự đoán một V_{pred} phản ánh các đặc điểm này.
- Hệ thống sau đó sẽ tìm kiếm trong cơ sở dữ liệu các phim có genres như ***"Action"*** hoặc ***"Science Fiction",*** overview_embedding gần với các khái niệm về "kỹ xảo", "người hùng", "tương lai", và có vote_average và popularity tương tự với những gì bạn đã xem.
- Do đó, một phim như ***"Dune"*** (nếu có trong data và có feature vector phù hợp) có thể được gợi ý vì nó có thể thao túng các thuộc tính về ***"khoa học viễn tưởng"***, ***"kỹ xảo hoành tráng"*** và được đánh giá cao, khớp với V_{pred}.

Tóm lại, mô hình LSTM học cách nắm bắt ***"sở thích theo thời gian"*** của người dùng thông qua các đặc trưng tổng hợp của phim, sau đó sử dụng những sở thích này để dự đoán "kiểu phim" tiếp theo và tìm kiếm phim phù hợp trong cơ sở dữ liệu.

3.2.7. Lưu và tích hợp mô hình

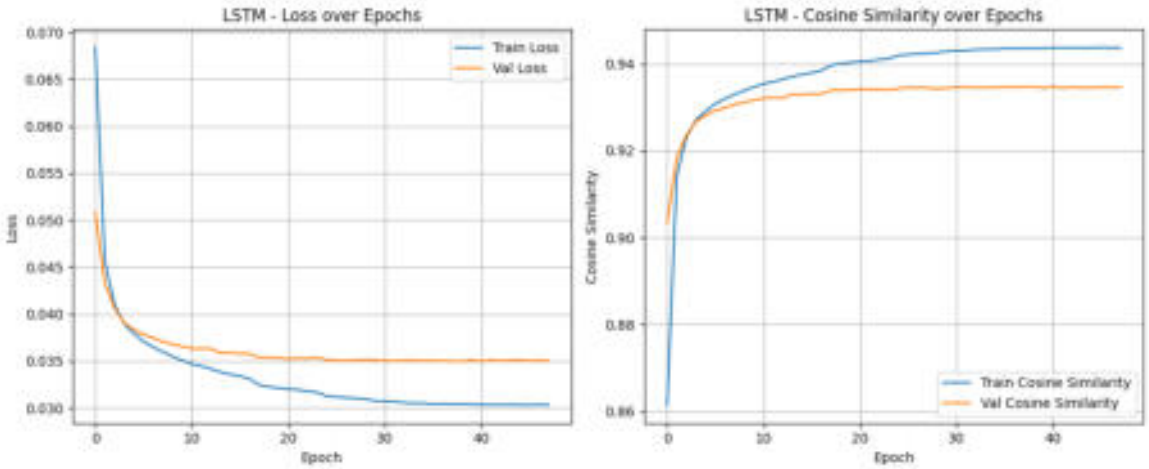
- Mô hình được lưu lại dạng .h5 và upload lên backend
- API /api/recommendations được xây dựng trong Flask để gọi mô hình
- Khi người dùng đặt đủ số lượng phim, hệ thống sinh gợi ý bằng cách:
    - Lấy danh sách các phim họ đã đặt
    - Dự đoán phim tiếp theo
    - Trả kết quả về frontend để hiển thị

3.3. Kết quả triển khai mô hình

Sau quá trình huấn luyện mô hình gợi ý phim sử dụng mạng LSTM kết hợp BERT embedding, hệ thống đã thu được những kết quả đáng khích lệ trên cả tập huấn luyện và tập kiểm thử.

3.3.1. Kết quả huấn luyện

- **Số epoch**: 50
- **Batch size**: 64
- **Split validation**: 80:20
- **Early stopping**: dừng sớm sau khi loss không cải thiện
- **Cosine Similarity đạt cao**:
    - Tập train: ~0.9439
    - Tập validation: ~0.9347
- **Loss thấp ổn định**:
    - Train loss: ~0.0303
    - Val loss: ~0.0350



Hình 3.1: Biểu đồ trực quan.

Hai biểu đồ Loss và Cosine Similarity theo số epoch cho thấy:

- Quá trình huấn luyện ổn định, không bị overfitting
- Cosine Similarity tăng đều và đạt ngưỡng tốt
- Loss giảm rõ rệt từ các epoch đầu tiên

Mô hình đã được lưu dưới dạng .h5 và đóng gói cùng API Flask để tích hợp vào backend.

3.3.2. Đánh giá mô hình

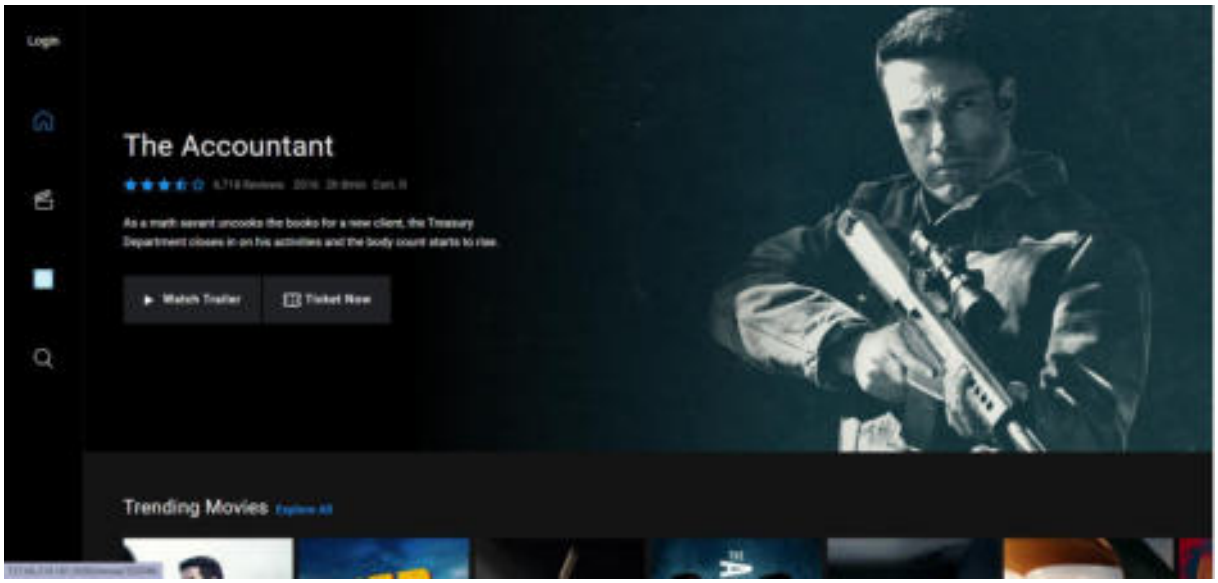
- **Ưu điểm**:
  - Hiệu suất cao với dữ liệu dạng sequence.
  - Có khả năng học được mối liên hệ giữa các phim người dùng đã xem.
  - Sử dụng embedding ngữ nghĩa từ BERT giúp hiểu rõ nội dung phim.
- **Hạn chế**:
  - Phụ thuộc vào độ chính xác của dữ liệu đầu vào từ TMDB.
  - Hệ thống vẫn có thể tạo gợi ý **khi người dùng đã đặt từ 1 phim trở lên**. Tuy nhiên, càng nhiều lịch sử đặt vé, độ chính xác và mức độ cá nhân hóa của gợi ý càng cao. Chưa áp dụng xử lý thời gian thực.

3.4. Kết quả triển khai ứng dụng

Sau khi hoàn thiện toàn bộ các module chức năng và mô hình AI, hệ thống đã được triển khai thành công trên VPS thực tế với địa chỉ IP: http://157.66.219.181/. Hệ thống hoạt động ổn định với đầy đủ các tính năng chính cho người dùng.

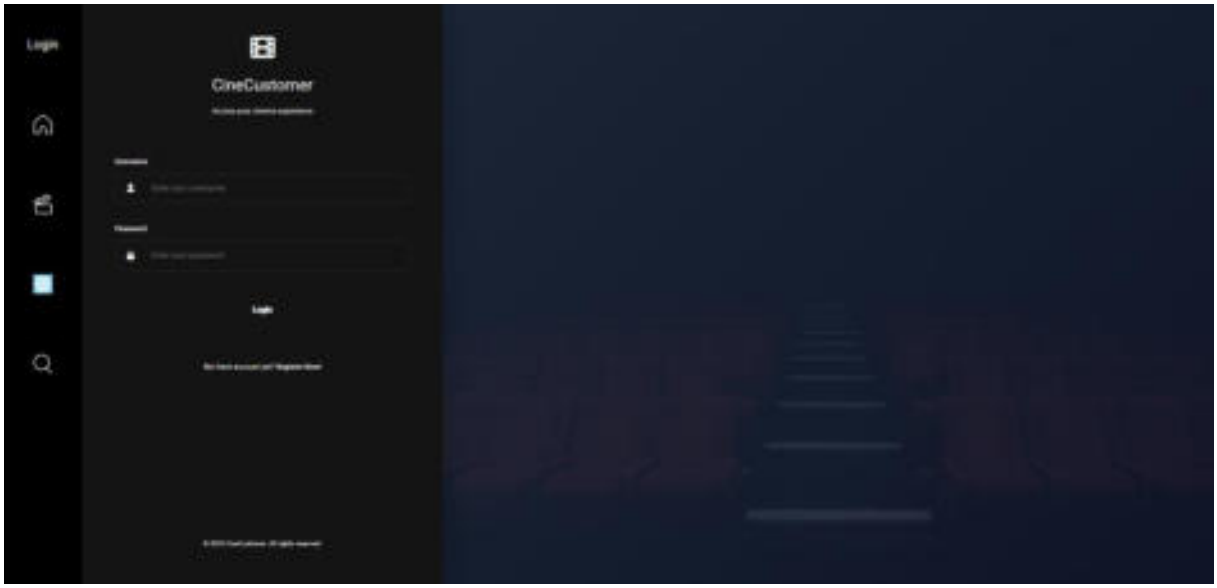
3.4.1 Màn hình giao diện người dùng

3.4.1.1. Màn hình trang chủ trang web



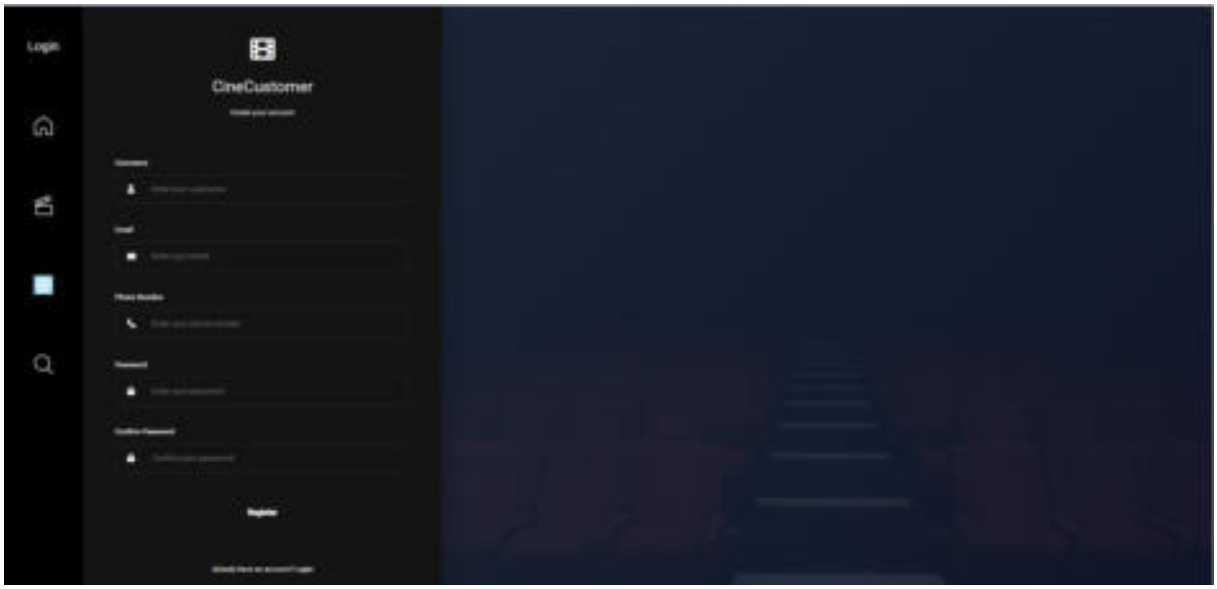
Hình 3.2: Màn hình trang chủ trang web.

3.4.1.2. Màn hình đăng nhập



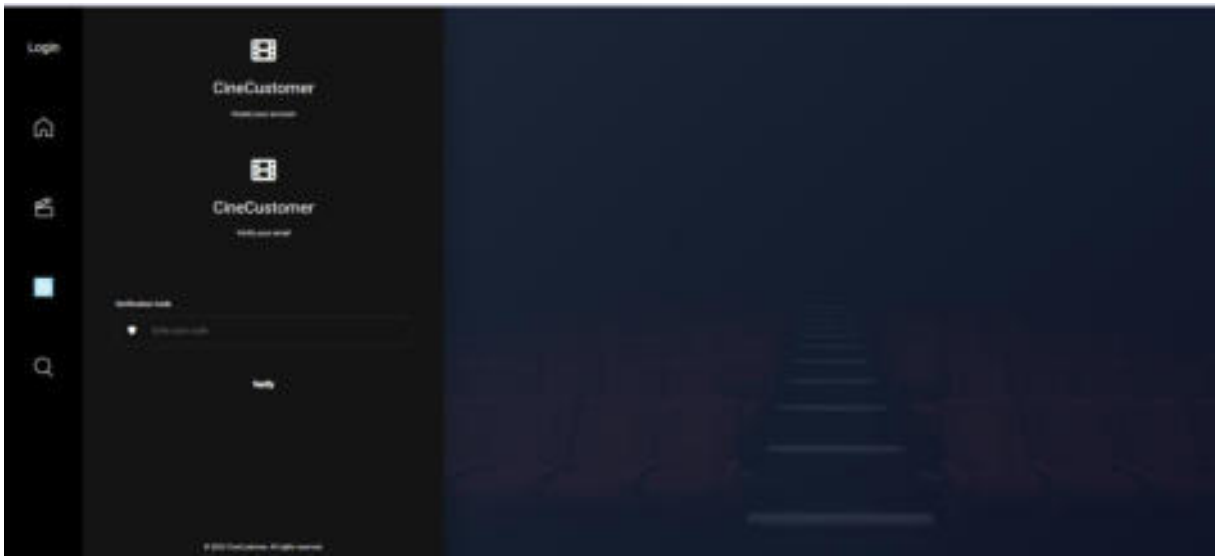
Hình 3.3: Màn hình đăng nhập.

3.4.1.3. Màn hình đăng ký



Hình 3. 4: Màn hình đăng ký tài khoản.

3.4.1.4. Màn hình xác thực email

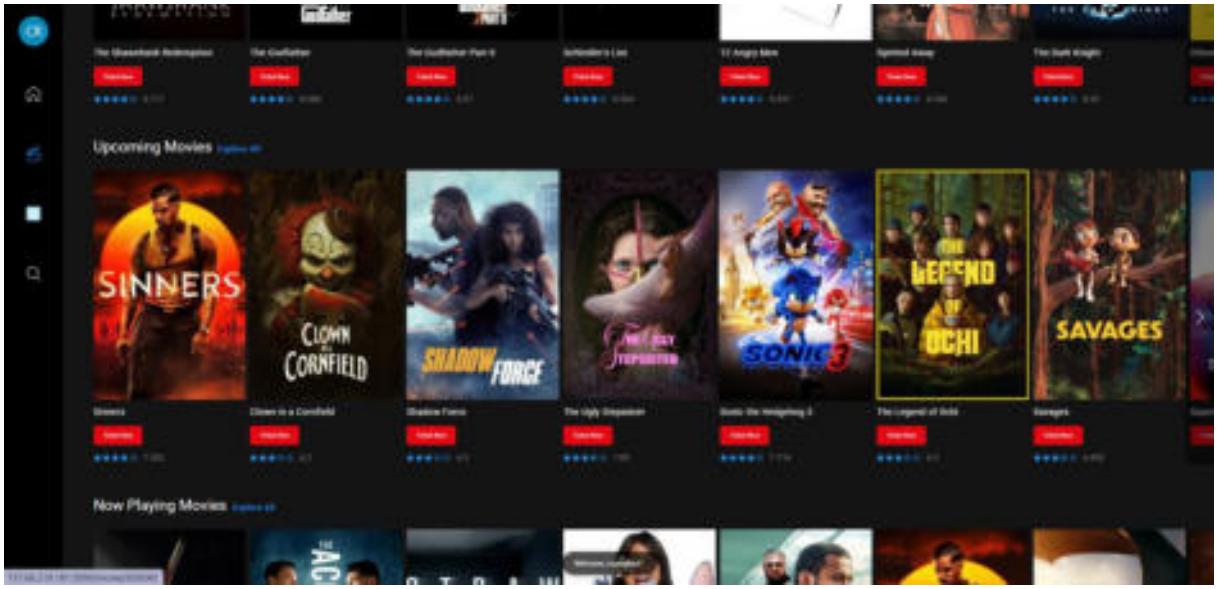


Hình 3. 5: Màn hình nhập code xác thực email.

3.4.1.5. Màn hình trang chủ sau khi người dùng đăng nhập

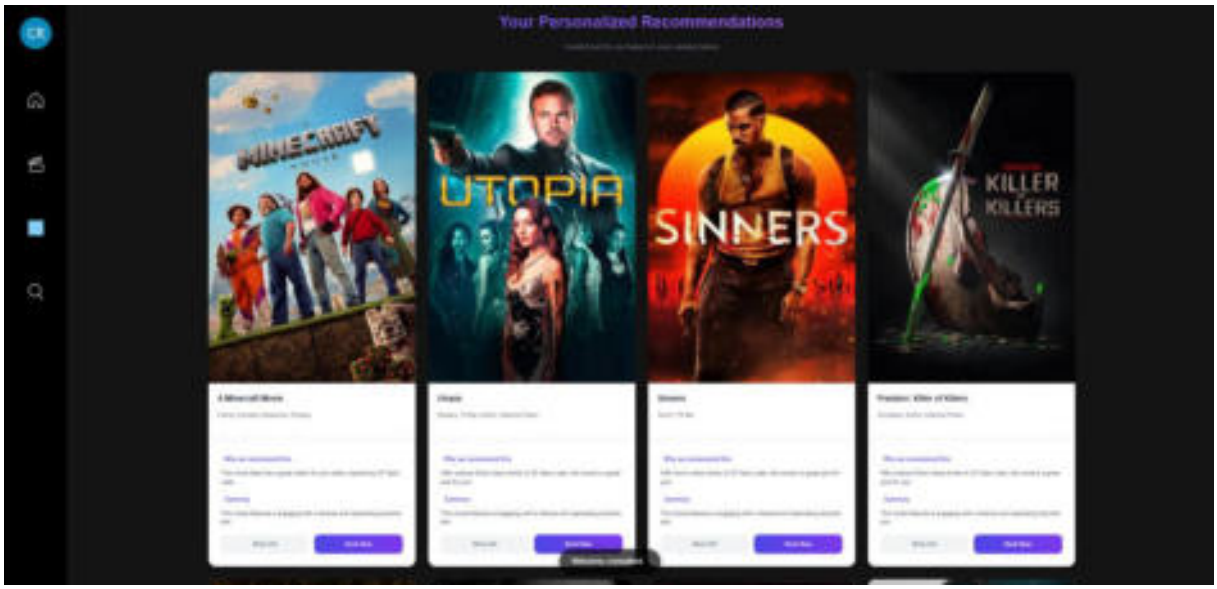
Hình 3. 6: Màn hình trang chủ sau khi người dùng đăng nhập.

3.4.1.6. Màn hình danh sách các loại phim



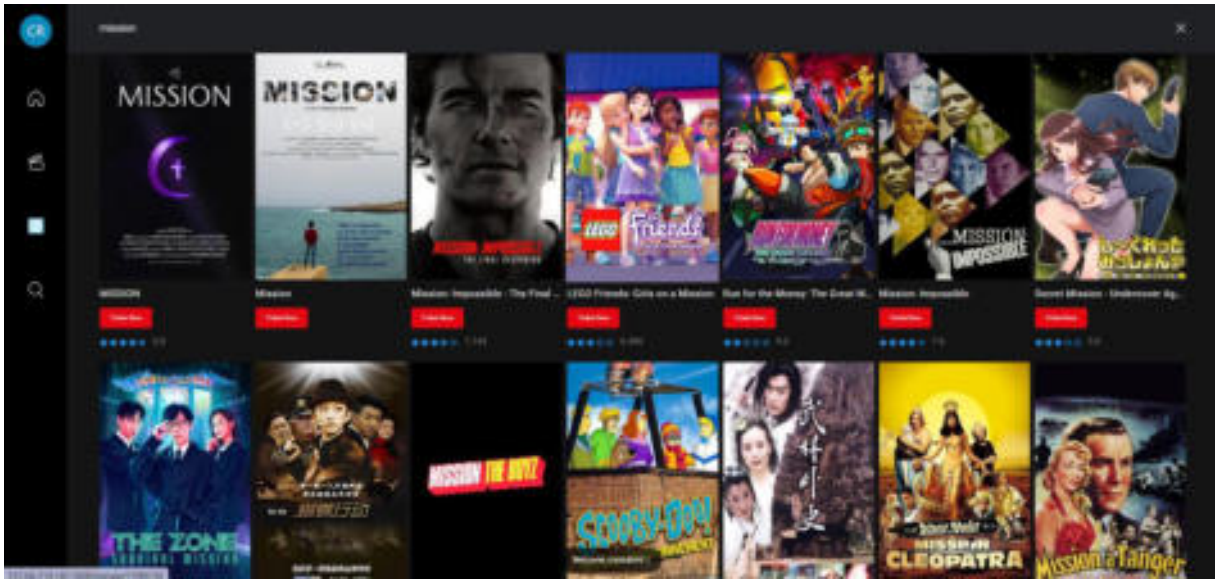
Hình 3. 7: Màn hình danh sách các thể loại phim.

3.4.1.7. Màn hình phim được gợi ý cho người dùng.



Hình 3. 8: Màn hình phim gợi ý cho người dùng.

3.4.1.8. Màn hình tìm kiếm phim

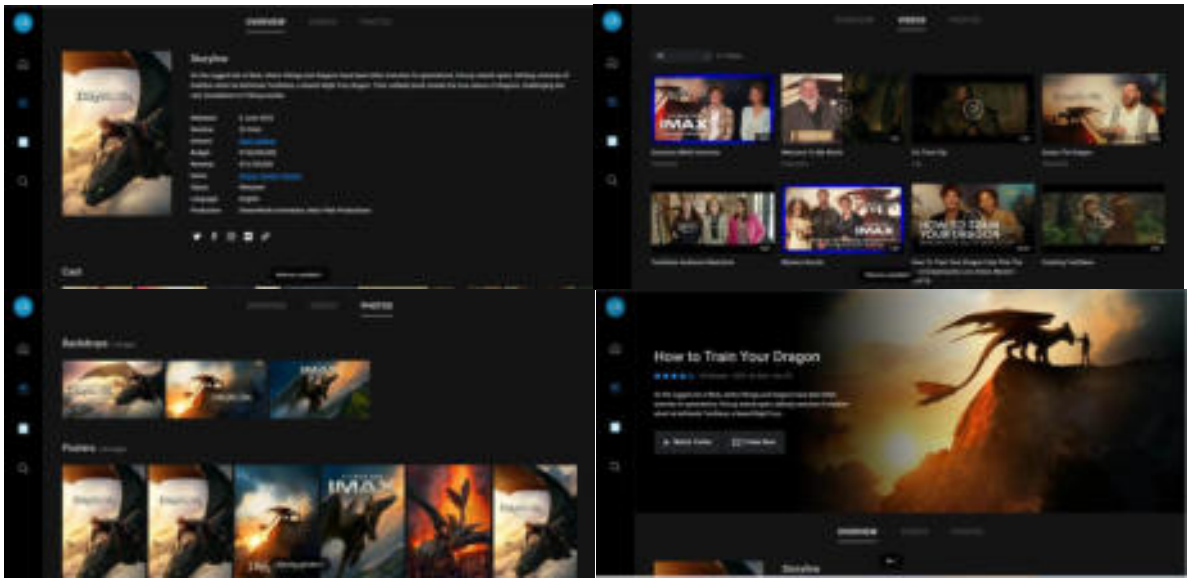


Hình 3. 9: Màn hình tìm kiếm phim.

3.4.1.9. Màn hình xem trailer phim

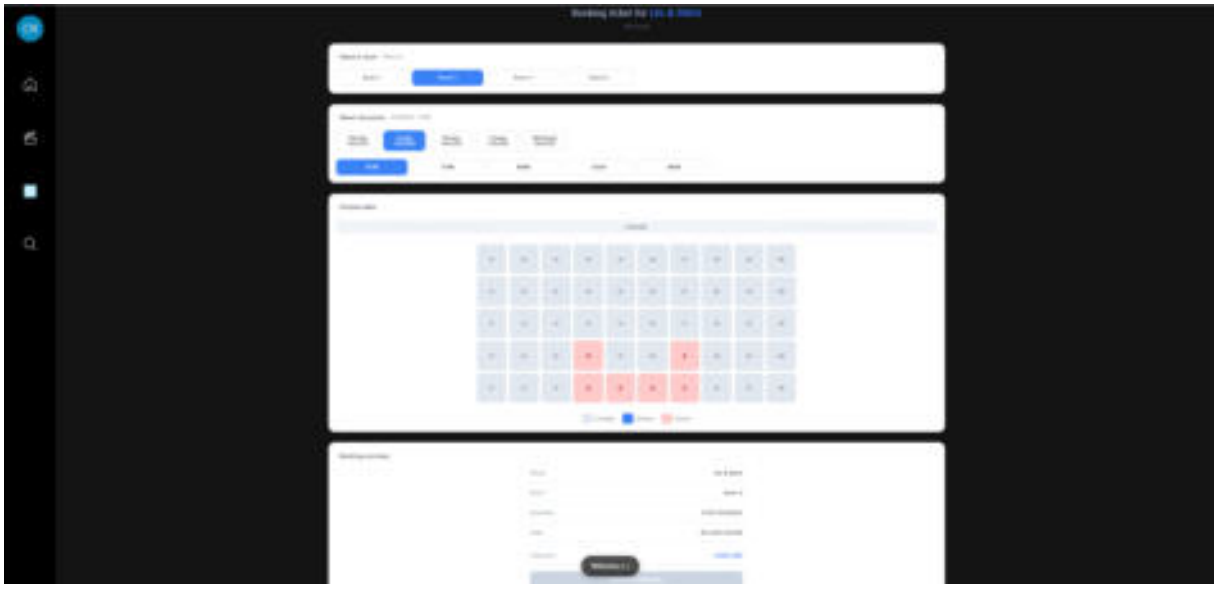
Hình 3.10: Màn hình xem trailer phim.

3.4.1.10. Màn hình xem chi tiết phim



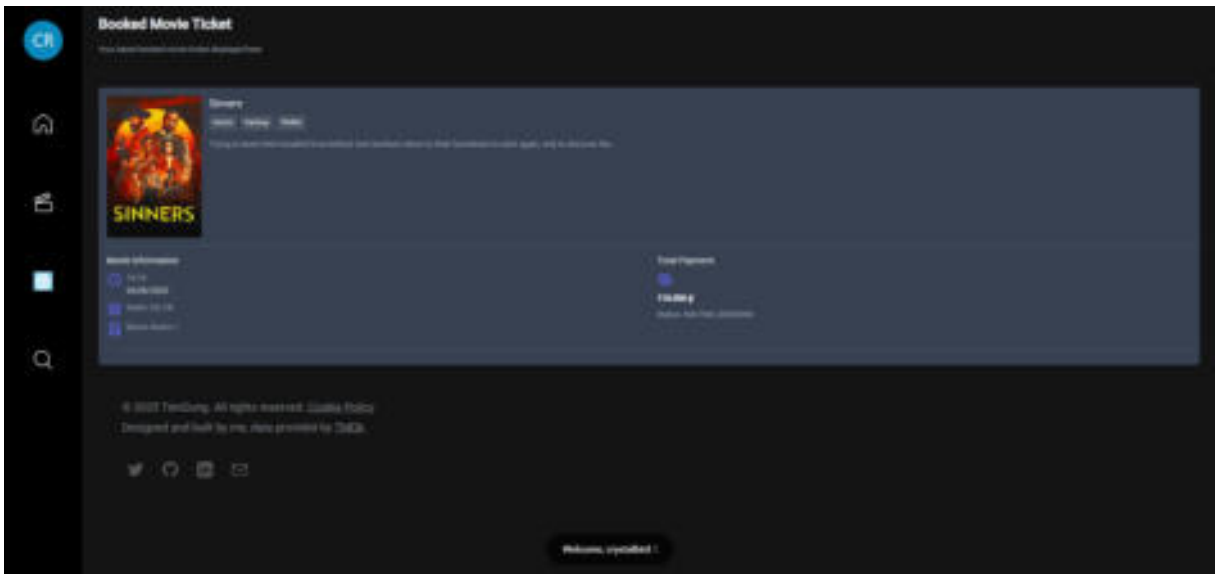
Hình 3. 11: Màn hình chi tiết phim

3.4.1.11. Màn hình đặt vé



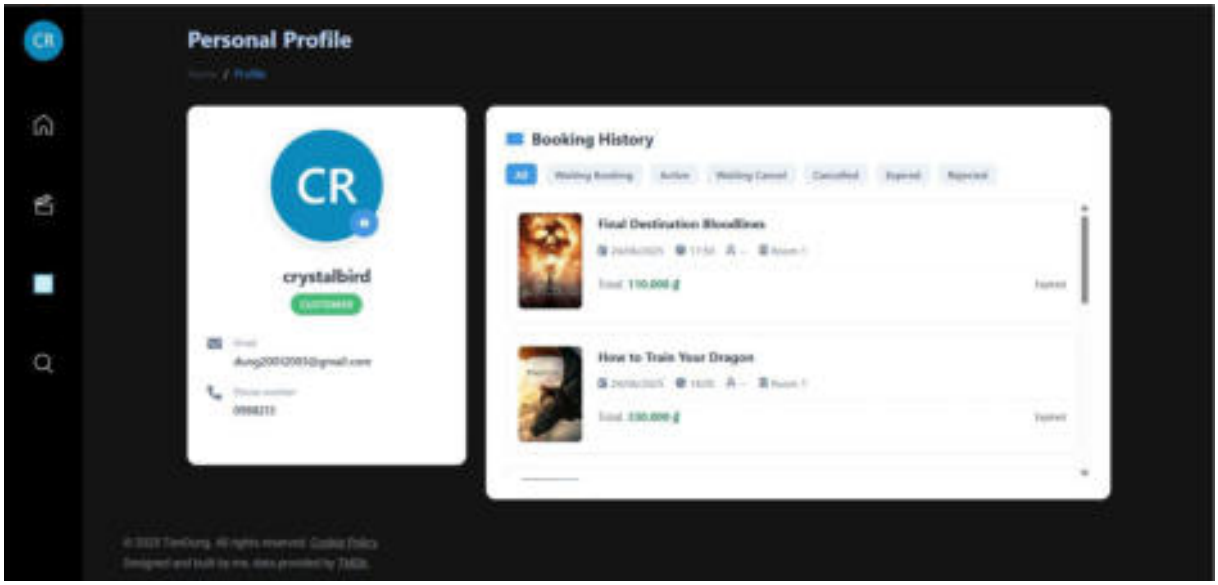
Hình 3. 12: Màn hình đặt vé xem phim.

3.4.1.12. Màn hình thông tin thanh toán



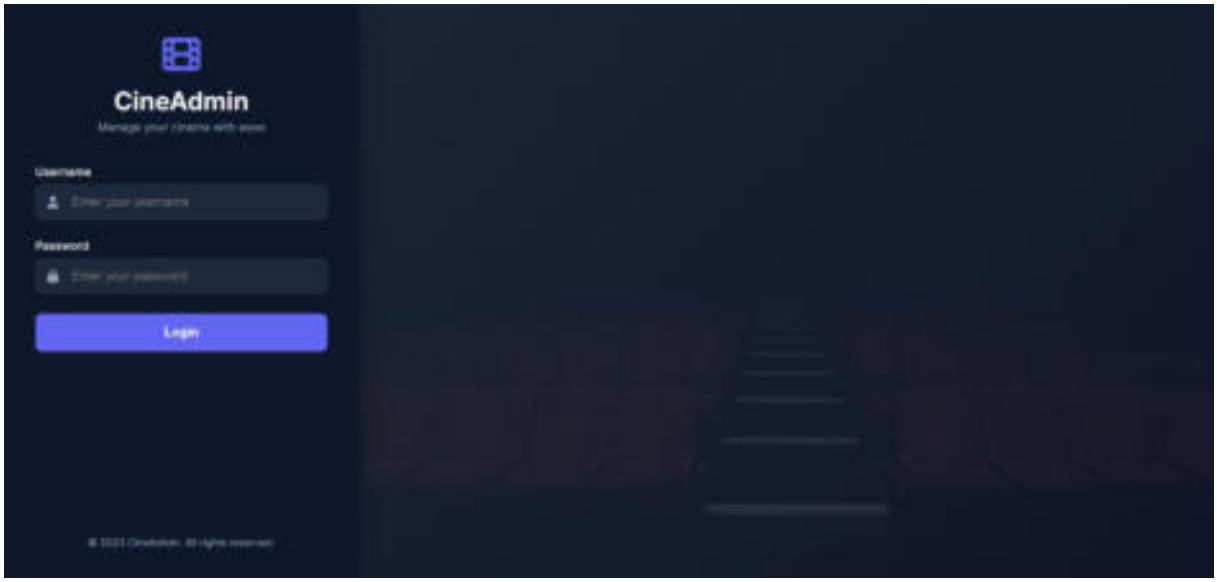
Hình 3.13: Màn hình thông tin thanh toán.

3.4.1.13. Màn hình thông tin cá nhân người dùng



Hình 3.14: Màn hình thông tin cá nhân người dùng.

3.4.1.14. Màn hình đăng nhập Admin

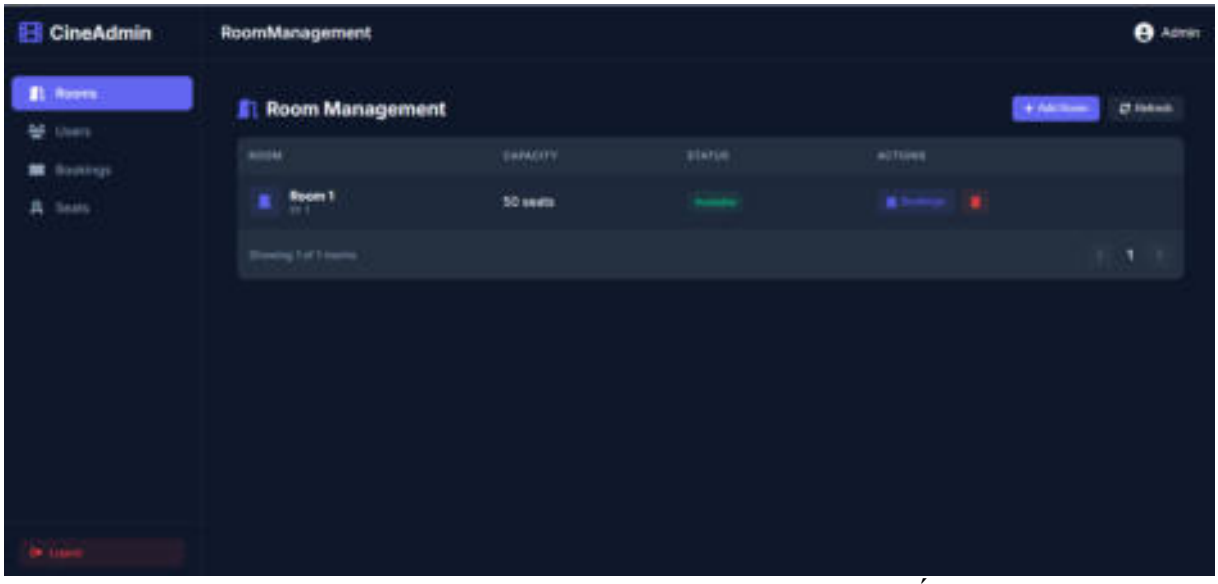


Hình 3.15: Màn hình đăng nhập admin.

3.4.1.15. Màn hình trang chủ Admin

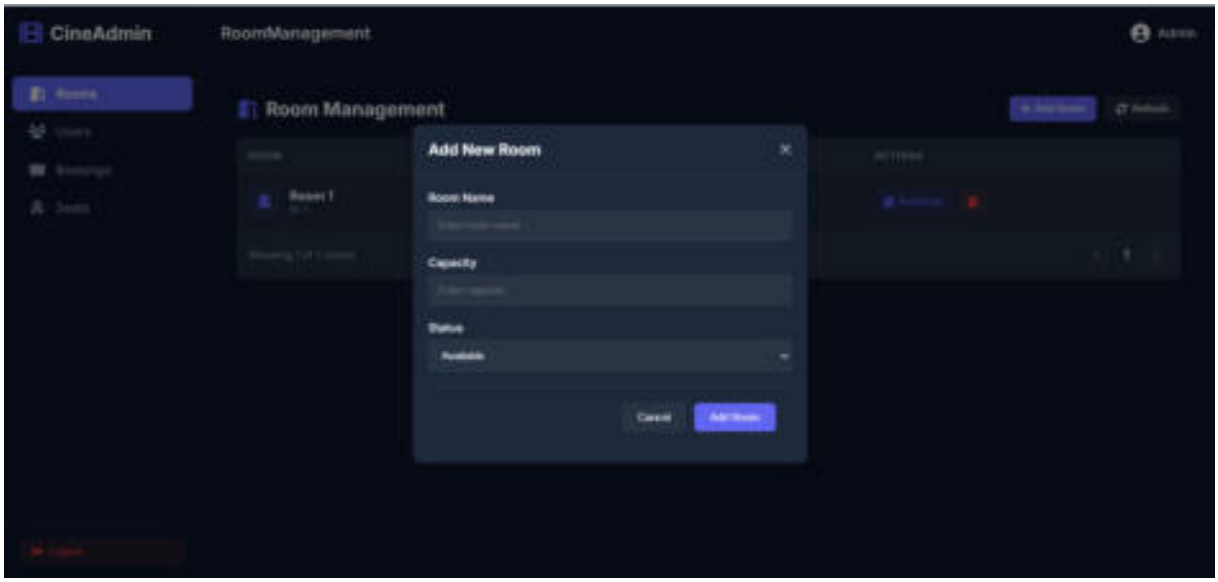
Hình 3.16: Màn hình trang chủ admin.

3.4.1.16: Màn hình xem danh sách phòng chiếu



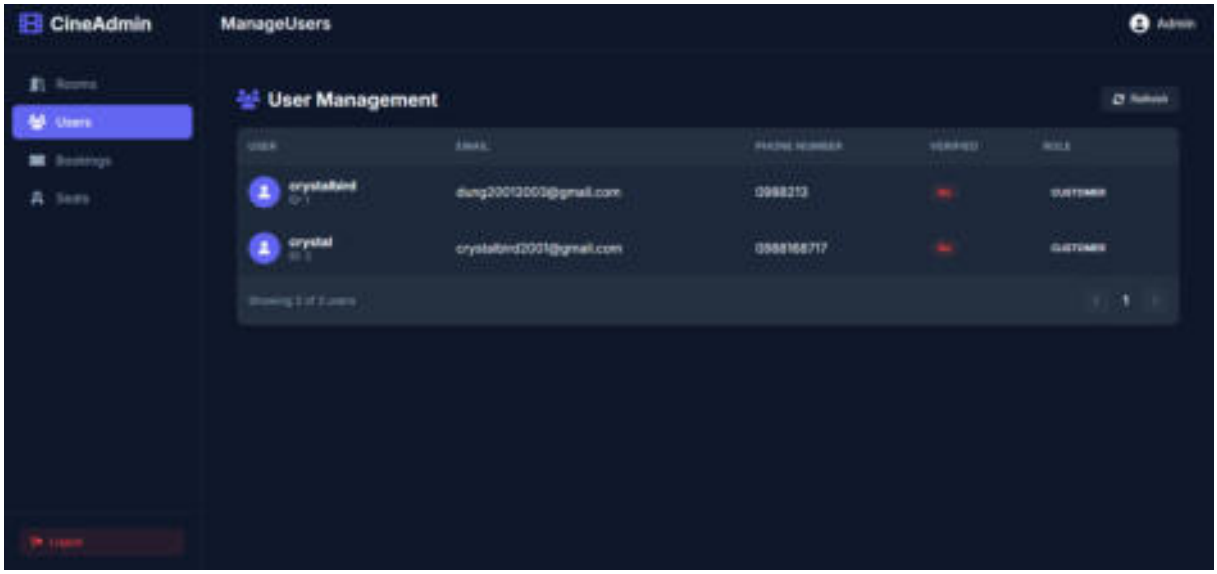
Hình 3.17: Màn hình xem danh sách phòng chiếu.

3.4.1.17. Màn hình chức năng thêm phòng



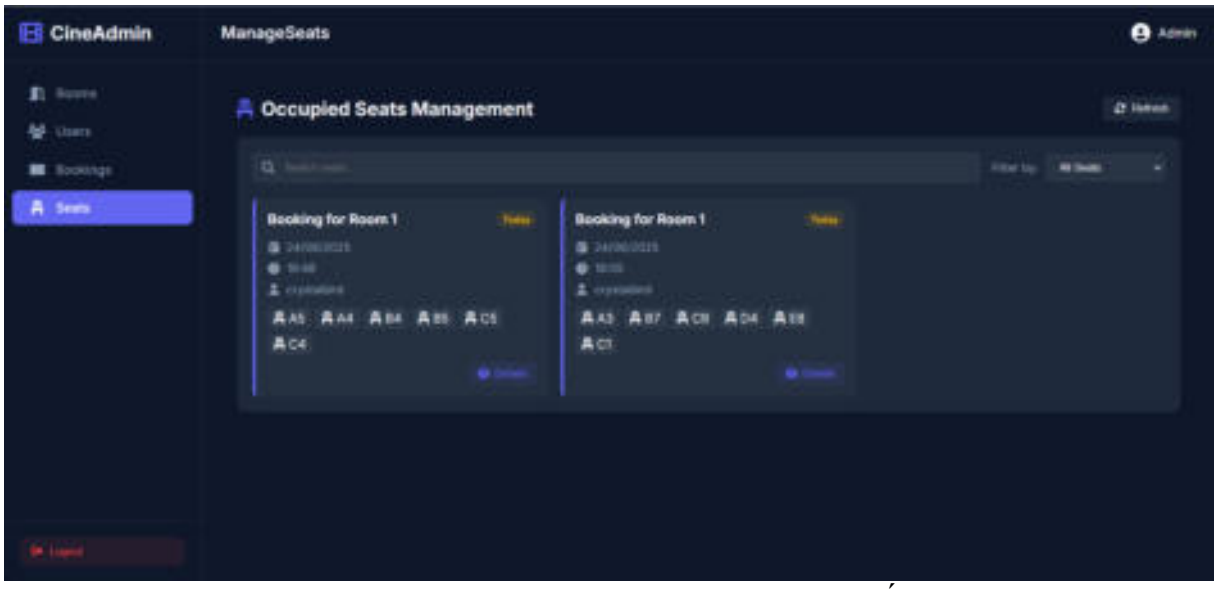
Hình 3.18: Màn hình chức năng thêm phòng.

3.4.1.18. Màn hình chức năng xem danh sách người dùng



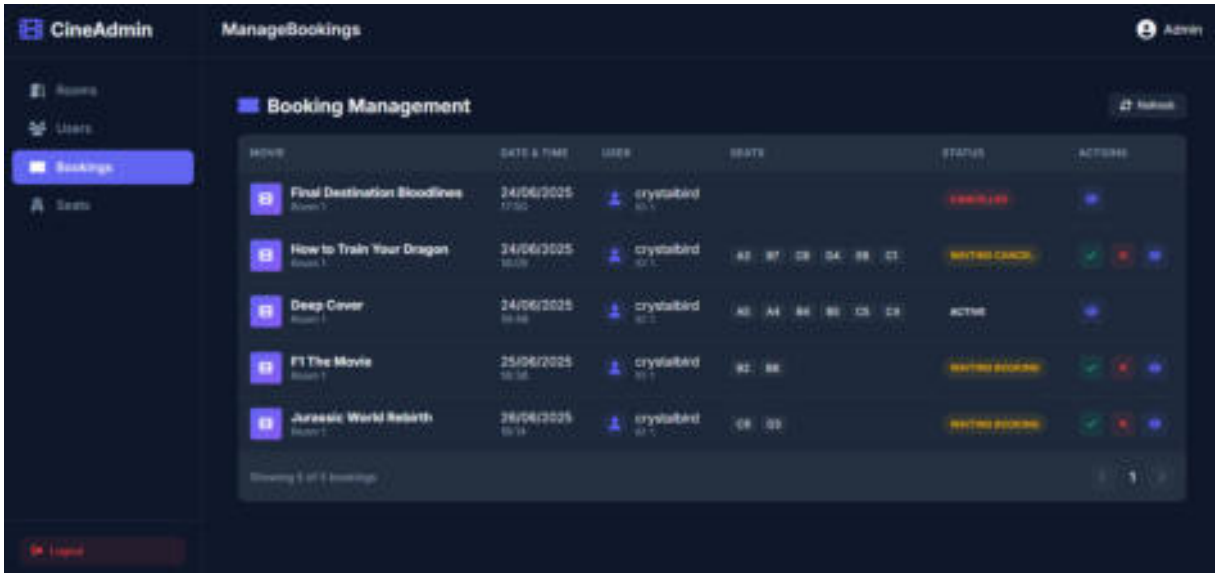
Hình 3.19: Màn hình chức năng xem danh sách người dùng.

3.4.1.19. Màn hình chức năng xem danh sách ghế được đặt



Hình 3.20: Màn hình chức năng xem danh sách ghế được đặt.

3.4.1.20. Màn hình chức năng xem danh sách vé đã được đặt



Hình 3.21: Màn hình chức năng xem danh sách vé được đặt.

3.5. Tự đánh giá kết quả triển khai hệ thống.

3.5.1. Ưu điểm

Bảng 3.8: Ưu điểm hệ thống.

Nhóm	Mô tả
1. Tính năng đầy đủ	Hệ thống tích hợp nhiều chức năng từ đăng ký, đăng nhập, đặt vé, xem lịch sử, xem trailer đến gợi ý phim thông minh.
2. Giao diện thân thiện	Frontend sử dụng Nuxt.js/Vue 2, giao diện dễ sử dụng, có hỗ trợ thông báo popup, kiểm tra trạng thái ghế động.
3. AI gợi ý hiệu quả	Mô hình LSTM kết hợp BERT giúp cá nhân hóa gợi ý phim theo lịch sử của từng người dùng. Cosine similarity cao, ổn định.
4. Tính bảo mật và xác thực tốt	Tích hợp JWT, mã hóa mật khẩu, xác thực email bằng mã gửi qua SMTP, mã lưu tạm thời trong Redis.
5. Triển khai hoàn chỉnh	Hệ thống được container hóa bằng Docker, deploy thành công trên VPS thật, hoạt động ổn định.
6. Khả năng mở rộng	Kiến trúc microservice tách biệt frontend, backend, AI giúp dễ nâng cấp và bảo trì.

3.5.2. Nhược điểm

Bảng 3.9: Nhược điểm hệ thống.

Nhóm	Mô tả
1. Gợi ý chưa theo thời gian thực	Mô hình AI cần tái huấn luyện khi dữ liệu thay đổi, chưa xử lý dynamic trên từng hành vi tức thời.
2. Dataset phụ thuộc TMDB	Hệ thống lệ thuộc nhiều vào API TMDB, có thể bị giới hạn nếu thay đổi chính sách API.
3. Thiếu CI/CD tự động	Việc deploy còn thủ công (git pull + docker-compose), chưa tích hợp pipeline tự động.
4. Quản lý lỗi còn đơn giản	Chưa có trang thông báo lỗi rõ ràng khi API gặp sự cố hoặc xác thực thất bại.

3.6. Kết chương

Chương này đã trình bày chi tiết quy trình triển khai và kiểm thử hệ thống **rạp chiếu phim tích hợp AI gợi ý phim**. Từ việc lựa chọn các công cụ phát triển phù hợp, tiến hành huấn luyện mô hình gợi ý bằng mạng LSTM kết hợp BERT, cho đến việc đóng gói, triển khai hệ thống bằng Docker trên VPS thực tế, tất cả các bước đều được thực hiện một cách bài bản và đồng bộ.

Mô hình AI đã cho kết quả khả quan với **Cosine Similarity đạt khoảng 0.9347**, cho thấy hiệu suất tốt và độ tương đồng cao trong việc gợi ý phim theo lịch sử người dùng. Ứng dụng frontend cũng được kiểm thử đầy đủ với các chức năng chính như đăng ký, đặt vé, xem trailer, lịch sử đặt vé và nhận gợi ý.

Thông qua quá trình triển khai thực tế, nhóm cũng đã ghi nhận các ưu điểm nổi bật như giao diện thân thiện, hệ thống gợi ý hiệu quả, bảo mật tốt và kiến trúc microservice rõ ràng. Tuy nhiên, một số hạn chế vẫn tồn tại như phụ thuộc vào TMDB API, chưa có CI/CD tự động và chưa có giao diện quản trị viên.

Nhìn chung, hệ thống đã hoàn thành đúng phạm vi và mục tiêu ban đầu đặt ra. Các kết quả đạt được là nền tảng quan trọng để nhóm tiếp tục cải tiến, mở rộng và phát triển trong tương lai.

KẾT LUẬN

1. Tổng hợp kết quả đạt được

Hệ thống **đặt vé xem phim tích hợp AI gợi ý phim** đã được triển khai thành công với các kết quả nổi bật:

- **Xây dựng thành công hệ thống web** gồm frontend (Nuxt.js), backend (Spring Boot) và mô hình AI (Flask).
- **Tích hợp đầy đủ tính năng người dùng**: đăng ký, đăng nhập, xác thực email, đặt vé, xem trailer, quản lý lịch sử vé và nhận gợi ý phim.
- **Gợi ý phim cá nhân hóa hiệu quả**, đạt Cosine Similarity ~ 0.9347 sau huấn luyện.
- **Triển khai thực tế bằng Docker trên VPS**, sử dụng MySQL và Redis.

2. Những vấn đề cần giải quyết

Mặc dù hệ thống đã vận hành đúng yêu cầu, một số vấn đề còn tồn tại cần được cải thiện:

- **Gợi ý chưa theo thời gian thực**: Mô hình chỉ đưa ra kết quả dựa trên dữ liệu huấn luyện tĩnh, chưa cập nhật theo các hành vi mới nhất của người dùng.
- **Triển khai còn thủ công**: Hệ thống chưa tích hợp CI/CD tự động hóa việc build và deploy.
- **Thiếu xử lý lỗi thân thiện**: Một số lỗi backend hiển thị dạng thô, chưa có UI bắt lỗi hoặc hiển thị thông báo rõ ràng cho người dùng.
- **Phụ thuộc TMDB API**: Việc thay đổi API từ phía TMDB có thể ảnh hưởng đến hệ thống.

3. Hướng phát triển trong tương lai

Để hệ thống trở nên hoàn thiện và có khả năng mở rộng thực tế hơn, cần xem xét các định hướng sau:

- **Cải tiến AI**: chuyển sang mô hình Transformer hoặc áp dụng các mô hình gợi ý hiện đại như Collaborative Filtering kết hợp Embedding.
- **Nâng cấp bảo mật và trải nghiệm xác thực**: thêm tính năng quên mật khẩu, OTP, xác thực 2 lớp.
- **Tích hợp CI/CD**: sử dụng GitHub Actions hoặc Jenkins để tự động hóa quy trình deploy.
- **Cá nhân hóa sâu hơn**: dựa trên cả lịch sử tìm kiếm, lượt xem trailer, đánh giá sao...
- **Thống kê & theo dõi**: hiển thị báo cáo doanh thu, số lượng vé, số người dùng... giúp mở rộng kinh doanh.

TÀI LIỆU THAM KHẢO

[1] The Movie Database (TMDB) API – Official Docs. [Online]. Available: https://developer.themoviedb.org/docs/getting-started . [Accessed: May, 2025]

[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). [Online]. Available: https://arxiv.org/abs/1810.04805 . [Accessed: May, 2025]

[3] Hochreiter, S., & Schmidhuber, J. (1997). **Long Short-Term Memory**. Neural Computation. [Online]. Available: https://www.bioinf.jku.at/publications/older/2604.pdf. [Accessed: May, 2025]

[4]Keras.[Online]. Available: https://keras.io/api/layers/recurrent_layers/lstm/.[Accessed: May, 2025]

[5] TensorFlow: An end-to-end open source machine learning platform. [Online]. Available: https://www.tensorflow.org. [Accessed: May, 2025]

[6] Scikit-learn: Cosine Similarity Function. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html. [Accessed: May, 2025]

[7] Sentence-Transformers Documentation. [Online]. Available: https://www.sbert.net/.[Accessed: May, 2025]

[8] PyTorch Official Documentation. [Online]. Available: https://pytorch.org/docs/stable/index.html. [Accessed: May, 2025]

[9] Transformers by Hugging Face. [Online]. Available: https://huggingface.co/transformers/.[Accessed: May, 2025]

[10] Google Colab Documentation. [Online]. Available: https://research.google.com/colaboratory/.[Accessed: May, 2025]

[11] Spring Boot Official Documentation. [Online]. Available: https://docs.spring.io/spring-boot/docs/current/reference/html/. [Accessed: May, 2025]

[12] Spring Security & JWT Authentication. [Online]. Available: https://www.baeldung.com/spring-security-oauth-jwt. [Accessed: May, 2025]

[13] Redis Official Documentation. [Online]. Available: https://redis.io/docs/. [Accessed: May, 2025]

[14] Docker Documentation. [Online]. Available: https://docs.docker.com/. [Accessed: May, 2025]

[15] Docker Compose Docs. [Online]. Available: https://docs.docker.com/compose/. [Accessed: May, 2025]

[16] Flask (Python Web Framework) Docs. [Online]. Available: https://flask.palletsprojects.com/. [Accessed: May, 2025]

[17] Pandas Library Documentation. [Online]. Available: https://pandas.pydata.org/docs/. [Accessed: May, 2025]

[18] NumPy Library Documentation. [Online]. Available: https://numpy.org/doc/. [Accessed: May, 2025]

[19] Vue.js Official Guide. [Online]. Available: https://vuejs.org/guide/introduction.html. [Accessed: May, 2025]

[20] Nuxt.js 2 Documentation. [Online]. Available: https://v2.nuxt.com/docs. [Accessed: May, 2025]

[21] Node.js Official Docs. [Online]. Available: https://nodejs.org/en/docs. [Accessed: May, 2025]

[22] Axios HTTP Client (JavaScript). [Online]. Available: https://axios-http.com/docs/intro. [Accessed: May, 2025]

[23] Gmail SMTP Settings for JavaMailSender. [Online]. Available: https://support.google.com/mail/answer/7126229?hl=en. [Accessed: May, 2025]

[24] JavaMailSender Spring Guide. [Online]. Available: https://www.baeldung.com/spring-email. [Accessed: May, 2025]
[25] JWT.io – JSON Web Token Introduction. [Online]. Available: https://jwt.io/introduction. [Accessed: May, 2025]
[26] MySQL Documentation. [Online]. Available: https://dev.mysql.com/doc/. [Accessed: May, 2025]
[27] GitHub Docs – Repository & Deployment. [Online]. Available: https://docs.github.com/en. [Accessed: May, 2025]
[28] GitHub Actions Documentation (CI/CD). [Online]. Available: https://docs.github.com/en/actions. [Accessed: May, 2025]
[29] Jenkins CI/CD Official Site. [Online]. Available: https://www.jenkins.io/doc/. [Accessed: May, 2025]
[30] Postman – API Testing Tool. [Online]. Available: ttps://www.postman.com/product/api-client/. [Accessed: May, 2025]
[31] Recurrent Neural Network. [Online]. Available: https://viblo.asia/p/recurrent-neural-networkphan-1-tong-quan-va-ung-dung-jvElaB4m5kw. [Accessed: Jun, 2025]
[32] What is vanishing gradient problem?[Online]. Available: https://www.engati.com/glossary/vanishing-gradient-problem#:~:text=Vanishing%20gradient%20problem%20is%20a,layers%20to%20the%20earlier%20layers. [Accessed: Jun, 2025]

[33] Vanishing and Exploding Gradients Problems in Deep Learning. Available: https://www.geeksforgeeks.org/deep-learning/vanishing-and-exploding-gradients-problems-in-deep-learning/. [Accessed: Jun, 2025]

[34] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," arXiv preprint arXiv:1301.3781, 2013. [Online]. Available: https://arxiv.org/abs/1301.3781. [Accessed: May 2025].
[35] Virtual Private Server. [Online]. Available: https://nhanhoa.com/tin-tuc/vps-la-gi-nhung-ai-nen-dung-vps.html. [Accessed: June 2025].

PHỤ LỤC