

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

This image shows a full page of a handwriting practice worksheet. It consists of approximately 20 horizontal rows. Each row is defined by two parallel dotted lines, creating a series of uniform gaps for letter height. The entire page is otherwise blank, with no margins, text, or other markings.

Trà Vinh, ngày tháng năm
Giáo viên hướng dẫn

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

This image shows a full page of primary-ruled paper. It features multiple sets of horizontal lines designed to guide young learners' handwriting. Each set consists of three lines: a solid top line, a dashed middle line, and a solid bottom line. These sets are repeated vertically down the entire page, providing ample space for practicing letter formation and alignment. The paper is otherwise blank, with no margins or additional markings.

Trà Vinh, ngày.... tháng....năm....

Thành viên hội đồng

(Ký và ghi rõ họ tên)

LỜI MỞ ĐẦU

Trong bối cảnh cuộc Cách mạng Công nghiệp 4.0 đang diễn ra mạnh mẽ, quá trình chuyển đổi số đã và đang len lỏi vào mọi khía cạnh của đời sống kinh tế - xã hội, trong đó không thể không kể đến ngành công nghiệp giải trí. Nhu cầu thưởng thức điện ảnh tại các rạp chiếu phim ngày càng trở nên phổ biến, tuy nhiên, quy trình mua vé truyền thống thường đi kèm với những bất tiện như tốn thời gian di chuyển, xếp hàng chờ đợi, và rủi ro hết vé cho các suất chiếu hấp dẫn. Những trở ngại này đã tạo ra một nhu cầu cấp thiết về một giải pháp hiện đại, tiện lợi và hiệu quả hơn.

Nhận thức được xu hướng và nhu cầu đó, nhóm chúng em đã quyết định thực hiện đề tài "Xây dựng Hệ thống Đặt vé Xem phim Trực tuyến - CineBooking" cho đồ án kết thúc học phần Công nghệ Phần mềm. Dự án này không chỉ là một ứng dụng web đơn thuần mà còn là sản phẩm của quá trình vận dụng tổng hợp các kiến thức, kỹ thuật và quy trình phát triển phần mềm hiện đại đã được học.

Báo cáo này sẽ trình bày một cách chi tiết toàn bộ quá trình thực hiện dự án, từ giai đoạn phân tích yêu cầu, thiết kế hệ thống, lựa chọn công nghệ, cho đến triển khai và kiểm thử sản phẩm. Nội dung báo cáo sẽ đi sâu vào việc áp dụng kiến trúc Client-Server, xây dựng các RESTful API, đóng gói ứng dụng bằng Docker, và tự động hóa quy trình CI/CD với GitHub Actions.

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành Thầy Nguyễn Bảo Ân, Giảng viên phụ trách môn học Công Nghệ Phần Mềm, người đã tận tình giảng dạy và định hướng cho chúng em trong suốt quá trình học tập và hoàn thành đồ án này. Nhờ sự truyền đạt kiến thức chuyên sâu, cùng với những góp ý và sự hỗ trợ tận tâm của Thầy, chúng em đã có cơ hội tiếp cận và hiểu sâu hơn về các phương pháp khai phá dữ liệu hiện đại, từ lý thuyết nền tảng về học máy cho đến các ứng dụng thực tiễn trong việc xử lý dữ liệu phức tạp, xây dựng và đánh giá các mô hình dự báo.

Chúng em cũng xin gửi lời cảm ơn đến quý Thầy Cô trong khoa Kỹ thuật và Công nghệ – Bộ môn Công nghệ thông tin, đã tạo ra một môi trường học tập và nghiên cứu khoa học, giúp chúng em trang bị được những kiến thức nền tảng vững chắc trong suốt quá trình theo học tại trường.

Mặc dù nhóm đã nỗ lực hết mình để hoàn thành bài báo cáo một cách nghiêm túc, nhưng do những hạn chế về mặt thời gian và kiến thức, sai sót là điều khó tránh khỏi. Chúng em rất mong nhận được những ý kiến đóng góp quý báu từ Thầy để đồ án được hoàn thiện hơn nữa.

Chúng em xin chân thành cảm ơn!

MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU	1
1.1. Tên Dự Án và Chủ Đề.....	1
1.2. Mục Tiêu Của Ứng Dụng.....	1
1.3. Lý Do Chọn Đề Tài	2
CHƯƠNG 2: PHÂN TÍCH YÊU CẦU	3
2.1. Yêu cầu chức năng	3
2.1.1. Chức năng dành cho Người dùng (User).....	3
2.1.2. Chức năng dành cho Quản trị viên (Admin)	4
2.2. Yêu cầu phi chức năng	5
CHƯƠNG 3: THIẾT KẾ HỆ THỐNG	6
3.1. Kiến trúc tổng thể	6
3.2. Thiết kế cơ sở dữ liệu	7
3.3. Thiết kế API	7
3.4. Thiết kế giao diện (UI/UX)	9
Chương 4: Triển Khai và Công Nghệ Sử Dụng.....	11
4.1. Công Nghệ Sử Dụng	11
4.1.1. Backend (Server-side)	11
4.1.2. Frontend (Client-side).....	11
4.1.3. Cơ sở dữ liệu.....	11
4.2. Quy Trình CI/CD với GitHub Actions.....	12
4.2.1. Mục tiêu	12
4.2.2. Quy trình CI.....	12
4.3. Cấu Hình Docker và Triển Khai Ứng Dụng.....	12
4.3.1. Cấu trúc Docker	13
4.3.2. Quy trình triển khai.....	13
4.4. Kết luận chương	14
CHƯƠNG 5: QUẢN LÝ DỰ ÁN	15
5.1. Mô hình phát triển phần mềm	15
5.2. Công cụ quản lý dự án: Jira.....	15
5.2.1. Product Backlog.....	15
5.2.2. Kế hoạch Sprint	16

5.3. Biểu đồ Burndown Chart.....	17
CHƯƠNG 6: KIỂM THỬ.....	18
6.1. Chiến lược kiểm thử	18
6.2. Công cụ kiểm thử	18
6.3. Kết quả kiểm thử API với Postman.....	18
6.4. Kiểm thử tự động với GitHub Actions.....	19
CHƯƠNG 7: ĐÁNH GIÁ VÀ KẾT LUẬN	21
7.1. Đánh giá kết quả đạt được	21
7.2. Những khó khăn gặp phải trong quá trình thực hiện.....	21
7.3. Bài học rút ra và đề xuất cải thiện trong tương lai	22
CHƯƠNG 8: PHỤ LỤC	23
8.1. Hướng dẫn cài đặt và chạy ứng dụng	23
8.2. Liên kết GitHub Repository	24

DANH MỤC CÁC HÌNH ẢNH

Hình 3. 1 Sơ đồ kiến trúc hệ thống	6
Hình 3. 2 Sơ đồ quan hệ thực thể.....	7
Hình 3. 3 thiết kế của Trang chủ.....	9
Hình 3. 4 thiết kế của Trang chi tiết phim	10
Hình 3. 5 thiết kế của Trang chọn ghế	10
Hình 4. 1 workflow đã chạy thành công tại tab "Actions"	12
Hình 4. 2 Triển khai backend lên Render.com	14
Hình 4. 3 Triển khai frontend lên Vercel.	14
Hình 5. 1 Product Backlog	16
Hình 5. 2 Burndown Chart của một Sprint	17
Hình 6. 1 request đặt vé thành công trên Postman.....	19
Hình 6. 2 GitHub Actions	20

CHƯƠNG 1: GIỚI THIỆU

1.1. Tên Dự Án và Chủ Đề

Tên dự án: CineBooking

Chủ đề: Xây dựng một ứng dụng web hoàn chỉnh cho phép người dùng xem thông tin phim, tra cứu lịch chiếu và thực hiện đặt vé xem phim trực tuyến.

1.2. Mục Tiêu Của Ứng Dụng

Đối với người dùng:

Cung cấp một nền tảng thân thiện, tiện lợi để tìm kiếm thông tin về các bộ phim đang chiếu và sắp chiếu.

Cho phép người dùng xem lịch chiếu chi tiết tại các cụm rạp khác nhau và thực hiện quy trình đặt vé nhanh chóng, bao gồm:

- Chọn suất chiếu
- Chọn ghế ngồi

Tự động gửi email xác nhận sau khi đặt vé thành công, giúp mang lại trải nghiệm chuyên nghiệp và đáng tin cậy.

Cung cấp chức năng quản lý tài khoản cá nhân và xem lại lịch sử vé đã đặt.

Đối với quản trị viên (Admin):

Xây dựng một khu vực quản trị an toàn để quản lý các tài nguyên cốt lõi của hệ thống, bao gồm:

- Phim
- Rạp
- Suất chiếu
- Người dùng
- Cung cấp trang tổng quan (Dashboard) để theo dõi các số liệu thống kê quan trọng của hệ thống.

Đối với mục tiêu học phần:

Áp dụng các kiến thức về quy trình phát triển phần mềm, đặc biệt là mô hình Agile/SCRUM.

Vận dụng kiến trúc Client-Server, tách biệt rõ ràng giữa:

- Frontend: Next.js
- Backend: NestJS

Sử dụng các công cụ bắt buộc trong quy trình phát triển phần mềm hiện đại như:

- Git, Docker, Swagger, Postman
- CI/CD với GitHub Actions

1.3. Lý Do Chọn Đề Tài

Trong bối cảnh công nghệ số phát triển mạnh mẽ, nhu cầu giải trí trực tuyến, đặc biệt là xem phim tại rạp, ngày càng tăng cao. Tuy nhiên, quy trình mua vé truyền thống thường tốn nhiều thời gian và công sức.

Việc xây dựng một hệ thống đặt vé trực tuyến không chỉ đáp ứng nhu cầu thực tế của thị trường, mà còn là một bài toán công nghệ thực tiễn, cho phép nhóm áp dụng và thực hành toàn diện các công nghệ hiện đại.

Đề tài này bao quát đầy đủ các khía cạnh của một ứng dụng web hoàn chỉnh:

- Xây dựng giao diện người dùng (UI/UX)
- Thiết kế và triển khai cơ sở dữ liệu
- Xây dựng API RESTful an toàn
- Đóng gói và tự động hóa quy trình triển khai

Đây là cơ hội giúp nhóm:

- củng cố kiến thức
- tích lũy kinh nghiệm thực tế
- chuẩn bị cho các dự án lớn trong tương lai

CHƯƠNG 2: PHÂN TÍCH YÊU CẦU

2.1. Yêu cầu chức năng

Đây là các chức năng cụ thể mà hệ thống phải cung cấp cho người dùng. Chúng được chia thành hai nhóm chính: chức năng dành cho người dùng thông thường và chức năng dành cho quản trị viên.

2.1.1. Chức năng dành cho Người dùng (User)

Xem danh sách phim

Hệ thống phải hiển thị danh sách các bộ phim dưới hai danh mục: "Phim Đang Chiếu" và "Phim Sắp Chiếu".

Mỗi bộ phim được hiển thị dưới dạng thẻ (card) bao gồm hình ảnh poster và tên phim.

Xem chi tiết phim

Khi người dùng nhấp vào một bộ phim, hệ thống phải hiển thị trang chi tiết bao gồm: mô tả, thể loại, thời lượng, trailer, và danh sách các suất chiếu trong ngày.

Đăng ký tài khoản

Người dùng có thể tạo một tài khoản mới bằng cách cung cấp họ tên, email và mật khẩu.

Hệ thống phải kiểm tra email đã tồn tại hay chưa và xác thực dữ liệu đầu vào.

Đăng nhập/Đăng xuất

- Người dùng có thể đăng nhập vào hệ thống bằng email và mật khẩu.
- Hệ thống sử dụng JWT để xác thực và duy trì phiên đăng nhập.
- Người dùng đã đăng nhập có thể đăng xuất khỏi tài khoản.

Đặt vé xem phim

- Sau khi chọn phim, người dùng có thể bắt đầu quy trình đặt vé.
- Người dùng có thể lọc suất chiếu theo ngày và cụm rạp.
- Hệ thống phải hiển thị sơ đồ ghế ngồi của phòng chiếu, cho biết ghế nào còn trống, ghế nào đã được đặt.
- Người dùng có thể chọn một hoặc nhiều ghế trống.
- Hệ thống sẽ hiển thị tổng số tiền tạm tính dựa trên số lượng ghế và giá vé của suất chiếu đó.

Xác nhận đặt vé

Sau khi hoàn tất đặt vé, hệ thống phải tự động gửi một email xác nhận đến địa chỉ email của người dùng, bao gồm các thông tin chi tiết: mã đặt vé, tên phim, rạp, suất chiếu, số ghế, và tổng tiền.

Xem lịch sử đặt vé

Người dùng đã đăng nhập có thể truy cập vào trang cá nhân để xem lại lịch sử các vé đã đặt thành công.

Tra cứu thông tin rạp và giá vé

Hệ thống cung cấp trang "Cụm Rạp" để người dùng xem danh sách các rạp theo từng tỉnh/thành phố.

Hệ thống cung cấp trang "Giá Vé" để người dùng tra cứu bảng giá vé chi tiết của từng rạp.

2.1.2. Chức năng dành cho Quản trị viên (Admin)

Đăng nhập với vai trò Admin

Hệ thống có cơ chế phân quyền, cho phép tài khoản có vai trò admin truy cập vào khu vực quản trị.

Xem trang tổng quan (Dashboard)

Admin có thể xem một trang tổng quan với các số liệu thống kê cơ bản về hệ thống (tổng số phim, suất chiếu, người dùng,...).

Quản lý Phim (CRUD)

- Admin có thể xem danh sách tất cả các bộ phim.
- Admin có thể thêm một bộ phim mới vào hệ thống.
- Admin có thể cập nhật thông tin của một bộ phim đã có.
- Admin có thể xóa một bộ phim khỏi hệ thống.

Quản lý Suất chiếu (CRUD)

- Admin có thể xem danh sách tất cả các suất chiếu.
- Admin có thể tạo một suất chiếu mới cho một bộ phim tại một phòng chiếu cụ thể.
- Admin có thể xóa một suất chiếu đã có.

Quản lý Rạp chiếu (CRUD)

Admin có thể xem, thêm, sửa, xóa thông tin các rạp chiếu phim.

Quản lý Người dùng

- Admin có thể xem danh sách tất cả người dùng trong hệ thống.

- Admin có thể xóa một tài khoản người dùng.

2.2. Yêu cầu phi chức năng

Tính Tin cậy (Reliability):

Hệ thống phải hoạt động ổn định, các chức năng cốt lõi như đặt vé và thanh toán phải được xử lý chính xác, tránh gây mất mát dữ liệu.

Tính Bảo mật (Security):

- Mật khẩu người dùng phải được mã hóa trước khi lưu vào cơ sở dữ liệu.
- Các API nhạy cảm (đặt vé, xem lịch sử, quản trị) phải được bảo vệ và yêu cầu xác thực JWT.
- Phải có cơ chế phân quyền rõ ràng giữa người dùng thường và admin.

Hiệu năng (Performance):

- Thời gian tải trang phải nhanh, dưới 3 giây cho các trang chính.
- Thời gian phản hồi của API phải thấp, đặc biệt là các API truy vấn dữ liệu.

Tính Dễ sử dụng (Usability):

Giao diện người dùng phải trực quan, thân thiện và dễ dàng thao tác trên cả máy tính và các thiết bị di động (thiết kế đáp ứng - responsive).

Tính Dễ bảo trì (Maintainability):

Mã nguồn phải được cấu trúc rõ ràng, tách biệt giữa frontend và backend. Việc chia nhỏ các chức năng thành các module (trong NestJS) và component (trong React) giúp dễ dàng sửa đổi và nâng cấp trong tương lai.

CHƯƠNG 3: THIẾT KẾ HỆ THỐNG

3.1. Kiến trúc tổng thể

Hệ thống CineBooking được xây dựng dựa trên **kiến trúc Client-Server** ba lớp (3-tier architecture), một mô hình phổ biến và hiệu quả cho các ứng dụng web hiện đại. Kiến trúc này giúp tách biệt rõ ràng các mối quan tâm, tăng tính bảo mật, khả năng mở rộng và dễ dàng bảo trì.

Lớp Trình diễn (Presentation Layer - Frontend):

- **Công nghệ:** Next.js (React) và Tailwind CSS.
- **Nhiệm vụ:** Chịu trách nhiệm hiển thị giao diện người dùng và xử lý các tương tác. Toàn bộ các trang mà người dùng nhìn thấy và thao tác (trang chủ, chi tiết phim, đặt vé,...) đều thuộc lớp này. Lớp này giao tiếp với Backend thông qua các lời gọi API RESTful.

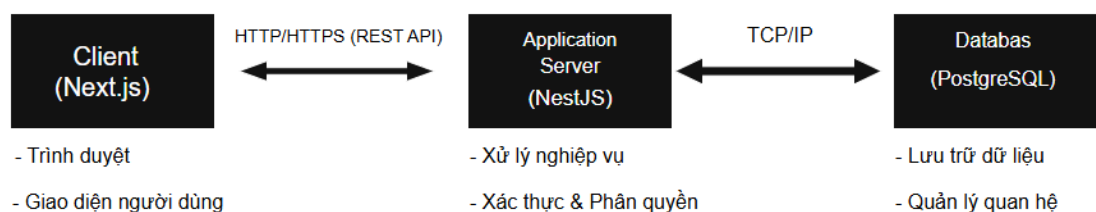
Lớp Ứng dụng (Application Layer - Backend):

- **Công nghệ:** NestJS (Node.js, TypeScript).
- **Nhiệm vụ:** Là "bộ não" của hệ thống, chịu trách nhiệm xử lý toàn bộ logic nghiệp vụ. Nó nhận yêu cầu từ Frontend, xử lý dữ liệu, tương tác với cơ sở dữ liệu, và trả về kết quả. Các chức năng như xác thực người dùng, kiểm tra ghế trống, tính toán giá vé, gửi email xác nhận đều được xử lý ở lớp này.

Lớp Dữ liệu (Data Layer - Database):

- **Công nghệ:** PostgreSQL.
- **Nhiệm vụ:** Chịu trách nhiệm lưu trữ và quản lý toàn bộ dữ liệu của ứng dụng một cách bền vững, bao gồm thông tin người dùng, phim, rạp, suất chiếu, và các đơn đặt vé.

Sơ đồ kiến trúc hệ thống:

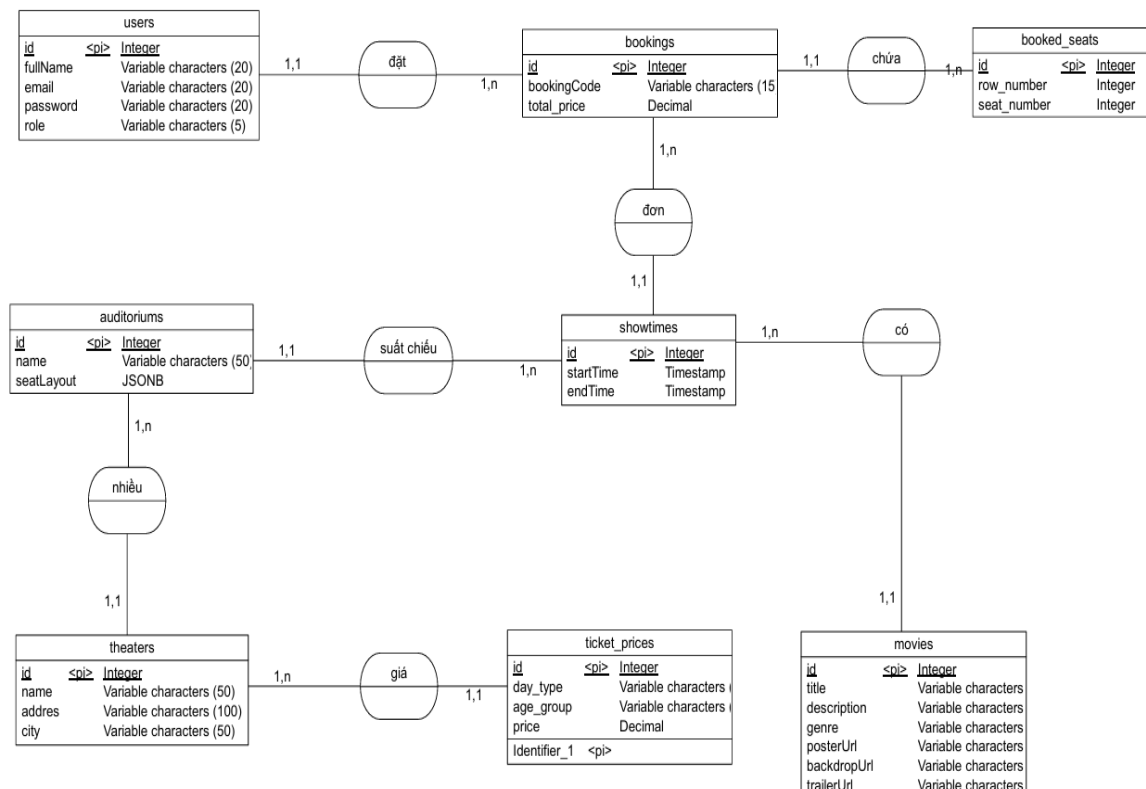


Hình 3. 1 Sơ đồ kiến trúc hệ thống

3.2. Thiết kế cơ sở dữ liệu

Cơ sở dữ liệu của hệ thống được thiết kế theo mô hình quan hệ để đảm bảo tính toàn vẹn và nhất quán của dữ liệu. Dưới đây là mô tả chi tiết các bảng (Entities) và mối quan hệ giữa chúng.

Sơ đồ quan hệ thực thể (ERD):



Hình 3. 2 Sơ đồ quan hệ thực thể

3.3. Thiết kế API

Hệ thống backend cung cấp một bộ các RESTful API để frontend tương tác nhằm phục vụ cho toàn bộ các chức năng tương tác giữa người dùng và hệ thống. Backend chịu trách nhiệm xử lý tất cả các logic nghiệp vụ quan trọng như:

- Quản lý thông tin phim, suất chiếu, rạp chiếu và người dùng
- Xử lý quy trình đặt vé và thanh toán
- Quản lý phân quyền, xác thực người dùng (JWT)

Toàn bộ tài liệu chi tiết và khả năng kiểm thử trực tiếp các API này cung cấp thông qua Swagger UI tại địa chỉ <https://cinebooking-backend.onrender.com/api-docs>

Mô tả các nhóm API chính:

Auth (/auth)

Xử lý các nghiệp vụ liên quan đến xác thực và đăng nhập.

- POST /auth/register: Đăng ký tài khoản người dùng mới.
- POST /auth/login: Đăng nhập vào hệ thống và nhận JWT token phục vụ việc xác thực.

Movies (/movies)

Cung cấp thông tin về phim đang chiếu và sắp chiếu.

- GET /movies: Lấy danh sách tất cả phim (có thể lọc theo trạng thái đang chiếu hoặc sắp chiếu).
- GET /movies/{id}: Lấy thông tin chi tiết về một bộ phim.
- GET /movies/{id}/showtimes: Lấy danh sách các suất chiếu của phim theo từng cụm rạp.

Theaters (/theaters)

Quản lý thông tin rạp chiếu và giá vé.

- GET /theaters: Lấy danh sách tất cả các rạp.
- GET /theaters/{id}/showtimes: Lấy lịch chiếu của rạp cụ thể theo ngày.
- GET /theaters/{id}/prices: Lấy bảng giá vé của mỗi rạp theo từng loại ghế và suất chiếu.

Bookings (/bookings)

Xử lý nghiệp vụ đặt vé xem phim.

- POST /bookings: Tạo một đơn đặt vé mới (yêu cầu xác thực người dùng).
- GET /bookings/my-history: Xem lịch sử đặt vé của người dùng đang đăng nhập (yêu cầu xác thực).

Admin (/admin)

Cung cấp các API cho chức năng quản trị (yêu cầu xác thực và quyền admin).

Quản lý Thống kê

GET /admin/stats: Lấy tổng hợp các số liệu thống kê (số lượng phim, suất chiếu, rạp, người dùng, ...).

Quản lý Phim

- POST /admin/movies: Thêm mới một bộ phim vào hệ thống.
- PUT /admin/movies/{id}: Cập nhật thông tin phim.
- DELETE /admin/movies/{id}: Xóa phim khỏi hệ thống.

Quản lý Suất chiếu

- GET /admin/showtimes: Lấy danh sách tất cả các suất chiếu (có thể lọc theo rạp và ngày).
- POST /admin/showtimes: Tạo mới một suất chiếu.
- DELETE /admin/showtimes/{id}: Xóa một suất chiếu.

Quản lý Rạp chiếu

- GET /admin/theaters: Lấy danh sách tất cả các rạp.
- POST /admin/theaters: Thêm mới một rạp chiếu vào hệ thống.
- PUT /admin/theaters/{id}: Cập nhật thông tin rạp.
- DELETE /admin/theaters/{id}: Xóa rạp khỏi hệ thống.

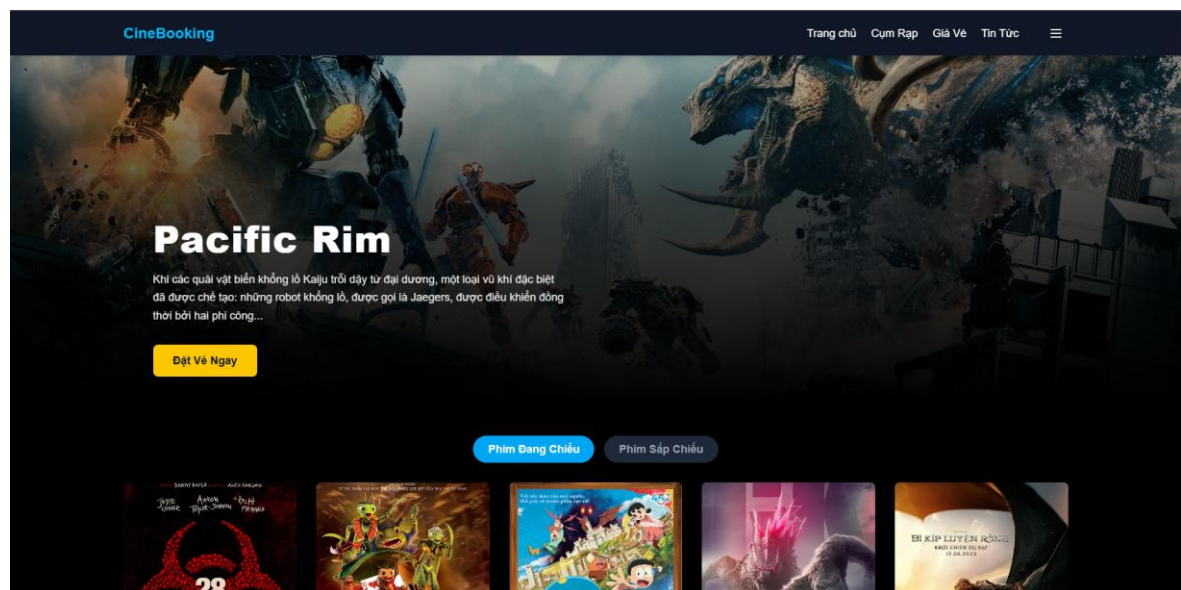
Quản lý Người dùng

- GET /admin/users: Lấy danh sách tất cả người dùng trong hệ thống.
- DELETE /admin/users/{id}: Xóa một tài khoản người dùng.

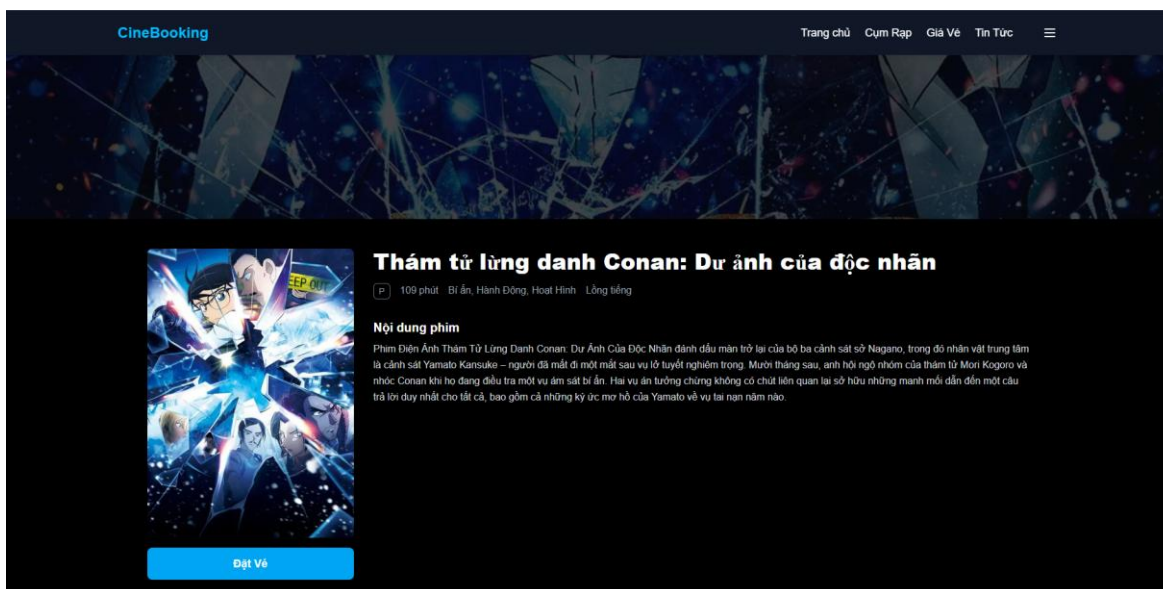
3.4. Thiết kế giao diện (UI/UX)

Giao diện người dùng của dự án được thiết kế bằng công cụ Figma, tập trung vào trải nghiệm người dùng thân thiện, trực quan và hiện đại. Thiết kế tuân thủ các nguyên tắc về sự nhất quán, dễ dàng điều hướng và có tính đáp ứng (responsive) để hoạt động tốt trên nhiều loại thiết bị.

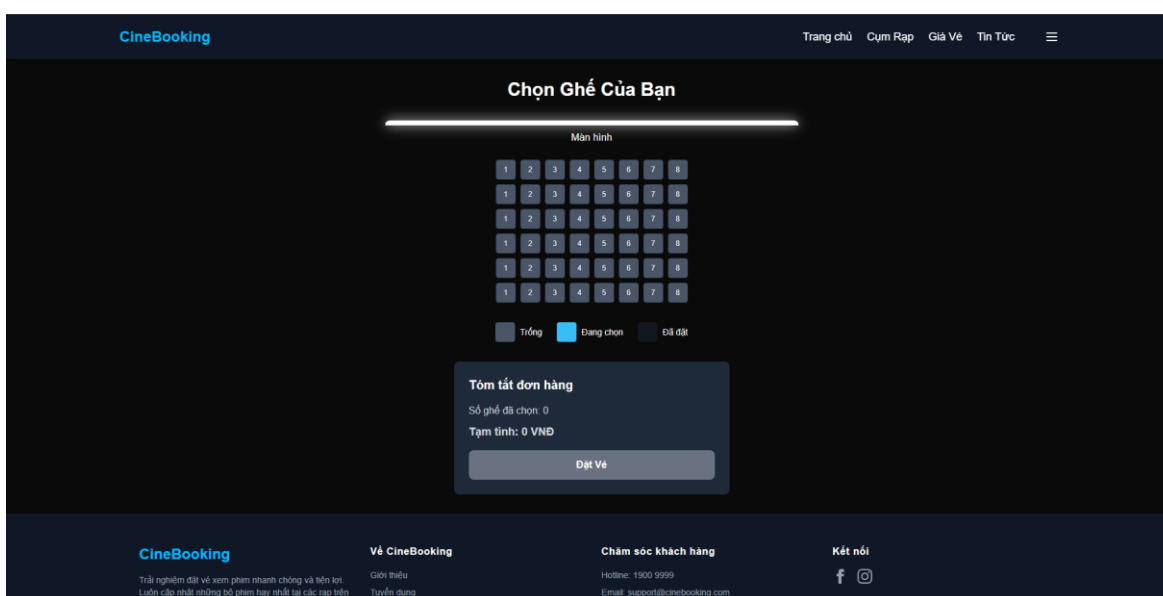
Liên kết đến file thiết kế Figma: <https://tinyurl.com/3yrachaa>



Hình 3. 3 thiết kế của Trang chủ



Hình 3. 4 thiết kế của Trang chi tiết phim



Hình 3. 5 thiết kế của Trang chọn ghế

Chương 4: Triển Khai và Công Nghệ Sử Dụng

4.1. Công Nghệ Sử Dụng

4.1.1. Backend (Server-side)

Công nghệ	Vai trò & Lý do lựa chọn
Node.js	Môi trường thực thi JavaScript trên máy chủ, có hiệu năng cao, cộng đồng lớn và phù hợp với các hệ thống real-time
TypeScript	Ngôn ngữ mở rộng của JavaScript, hỗ trợ kiểm tra kiểu tĩnh, giúp giảm lỗi runtime và tăng khả năng bảo trì.
NestJS	Framework Node.js hiện đại, hỗ trợ cấu trúc module hóa rõ ràng, dễ mở rộng và tích hợp tốt với TypeScript.
TypeORM	ORM giúp thao tác với cơ sở dữ liệu thông qua các thực thể (Entity), thay thế cho việc viết SQL thuần.
Passport.js	Thư viện xác thực người dùng linh hoạt, hỗ trợ xác thực qua JWT.
Bcrypt	Thư viện băm mật khẩu an toàn, giúp bảo mật dữ liệu người dùng.
Class-validator	Xác thực dữ liệu đầu vào thông qua decorator, đảm bảo tính toàn vẹn của thông tin truyền vào hệ thống.

4.1.2. Frontend (Client-side)

Công nghệ	Vai trò & Lý do lựa chọn
React	Thư viện xây dựng giao diện người dùng phổ biến nhất hiện nay, dễ tái sử dụng và mở rộng.
Next.js	Framework nâng cao của React, hỗ trợ SSR (Server-side Rendering), cải thiện tốc độ tải trang và SEO tốt.
Tailwind CSS	Framework CSS theo hướng utility-first, giúp xây dựng UI nhanh chóng, đồng nhất và dễ bảo trì.

4.1.3. Cơ sở dữ liệu

Công nghệ	Vai trò & Lý do lựa chọn
PostgreSQL	Hệ quản trị cơ sở dữ liệu quan hệ mạnh mẽ, mã nguồn mở, hỗ trợ tính năng ACID đảm bảo độ an toàn cho giao dịch.

4.2. Quy Trình CI/CD với GitHub Actions

Dự án áp dụng mô hình CI/CD (Continuous Integration / Continuous Deployment) để tự động hóa toàn bộ quy trình kiểm thử và triển khai mỗi khi có thay đổi trên hệ thống.

4.2.1. Mục tiêu

- Đảm bảo chất lượng mã nguồn: Kiểm tra tự động để phát hiện lỗi sớm.
- Tự động hóa quy trình build: Giảm thiểu lỗi do thao tác thủ công.
- Rút ngắn thời gian triển khai: Tự động hóa các bước build, test, lint.

4.2.2. Quy trình CI

Mỗi khi có push hoặc pull request vào nhánh main, hệ thống sẽ tự động thực hiện:

1. Kiểm tra mã nguồn
 - Chạy npm run lint để kiểm tra quy chuẩn code với ESLint.
2. Cài đặt môi trường
 - Cài đặt Node.js (phiên bản 20.x cho backend, 18.x cho frontend).
3. Cài đặt dependencies
 - Chạy npm install để cài đặt các thư viện cần thiết.
4. Build dự án
 - Chạy npm run build để đảm bảo dự án có thể biên dịch thành công.
5. Thông báo lỗi nếu có
 - Nếu bất kỳ bước nào thất bại, quá trình CI sẽ dừng lại và báo lỗi trên GitHub Actions.



Hình 4. 1 workflow đã chạy thành công tại tab "Actions"

4.3. Cấu Hình Docker và Triển Khai Ứng Dụng

Hệ thống CineBooking được đóng gói toàn bộ bằng Docker nhằm đảm bảo:

- Tính nhất quán giữa môi trường phát triển và production
- Triển khai nhanh chóng và dễ dàng di chuyển môi trường

4.3.1. Cấu trúc Docker

docker-compose.yml:

Quản lý 3 dịch vụ chính:

Dịch vụ	Mô tả
db	Dịch vụ cơ sở dữ liệu PostgreSQL
backend_service	API NestJS (Server)
frontend_service	Giao diện người dùng Next.js

Dockerfile riêng cho từng project:

- backend/Dockerfile để build NestJS API
- frontend/Dockerfile để build Next.js App\

4.3.2. Quy trình triển khai

Môi trường	Triển khai	Mô tả
Local Development	docker-compose up	Khởi tạo toàn bộ hệ thống trên máy cá nhân
Production (Cloud)	Render.com & Vercel	

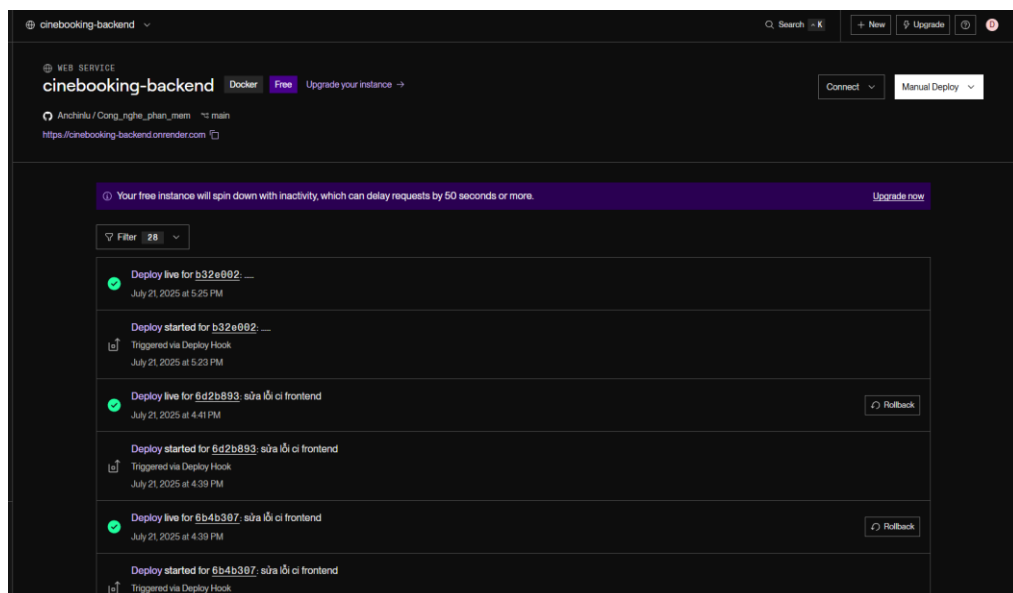
Chi tiết triển khai

Backend & Database:

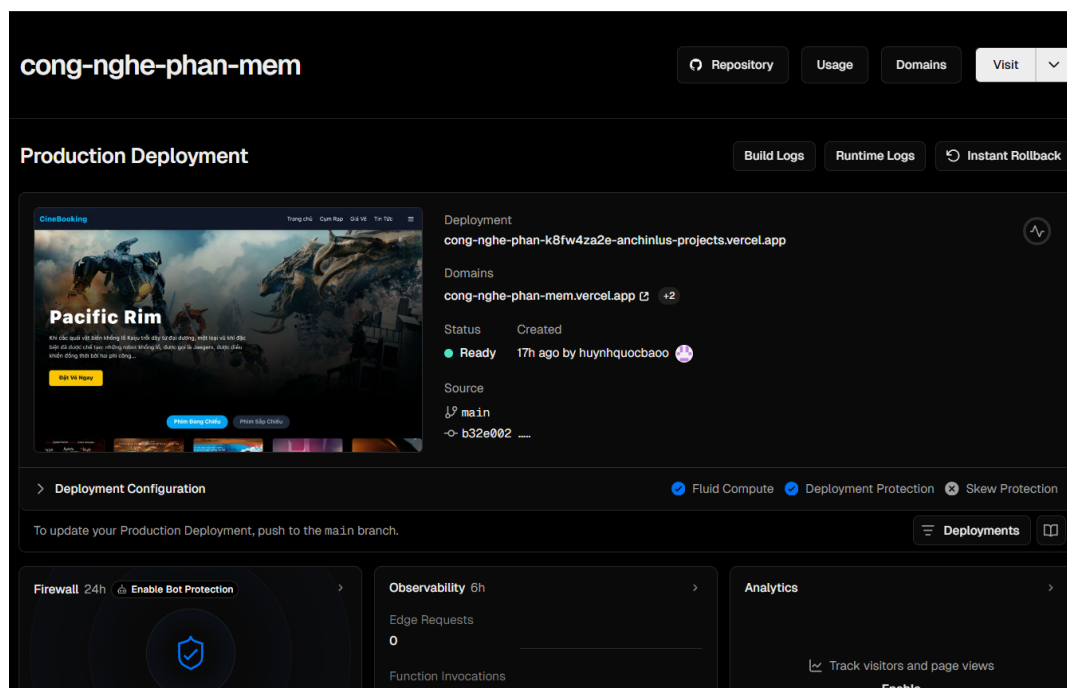
- Triển khai backend lên Render.com
- Render tự động đọc Dockerfile, build image và chạy service.
- PostgreSQL được cung cấp trực tiếp bởi Render với chính sách backup tự động.

Frontend:

- Triển khai frontend lên Vercel.
- Vercel kết nối với GitHub, tự động build và deploy khi có thay đổi
- Kết nối giữa frontend & backend:
- Biến môi trường NEXT_PUBLIC_API_URL được sử dụng để chỉ định địa chỉ API động, phù hợp với môi trường dev và production.



Hình 4. 2 Triển khai backend lên Render.com



Hình 4. 3 Triển khai frontend lên Vercel.

4.4. Kết luận chương

Việc áp dụng các công nghệ hiện đại cùng quy trình CI/CD và Docker đã giúp nhóm phát triển nhanh chóng, dễ dàng kiểm soát chất lượng, đồng thời đảm bảo việc triển khai diễn ra thuận lợi trên các nền tảng cloud. Điều này góp phần tạo ra một hệ thống đồng nhất, linh hoạt và sẵn sàng mở rộng trong tương lai.

CHƯƠNG 5: QUẢN LÝ DỰ ÁN

5.1. Mô hình phát triển phần mềm

Để đảm bảo dự án được thực hiện một cách linh hoạt, hiệu quả và có khả năng thích ứng cao với các thay đổi, nhóm đã quyết định áp dụng mô hình phát triển phần mềm Agile, cụ thể là phương pháp luận Scrum.

Scrum là một khung làm việc (framework) giúp các nhóm phát triển các sản phẩm phức tạp một cách hiệu quả. Các nguyên tắc cốt lõi của Scrum được áp dụng trong dự án bao gồm:

Phát triển lặp (Iterative Development): Dự án được chia thành các chu kỳ phát triển ngắn, gọi là Sprint, mỗi Sprint thường kéo dài 2-3 tuần.

Tăng trưởng (Incremental Growth): Sau mỗi Sprint, nhóm sẽ tạo ra một phiên bản phần mềm có thể hoạt động được và có thêm các chức năng mới.

Tự quản (Self-organizing): Các thành viên trong nhóm tự tổ chức và phân công công việc để đạt được mục tiêu của Sprint.

Hợp tác và Minh bạch: Toàn bộ tiến độ, kế hoạch và các vấn đề được theo dõi một cách minh bạch thông qua công cụ quản lý dự án.

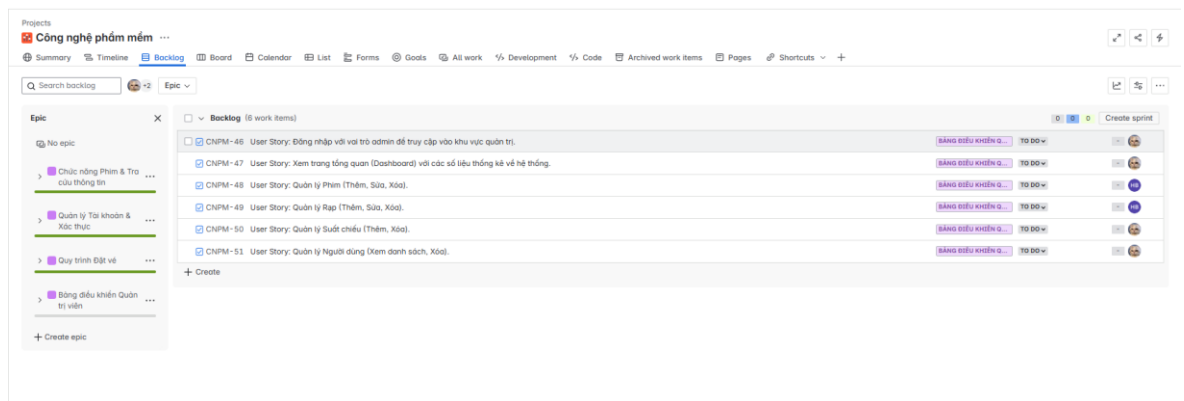
5.2. Công cụ quản lý dự án: Jira

Để triển khai mô hình Scrum một cách hiệu quả, nhóm đã sử dụng Jira Software làm công cụ quản lý dự án chính. Jira cung cấp đầy đủ các tính năng cần thiết để:

- Xây dựng và quản lý Product Backlog.
- Lập kế hoạch và theo dõi tiến độ của từng Sprint.
- Phân công và quản lý các nhiệm vụ (tasks) cho từng thành viên.
- Tự động tạo các báo cáo trực quan như Burndown Chart để theo dõi tiến độ thực tế so với kế hoạch.

5.2.1. Product Backlog

Product Backlog là một danh sách được sắp xếp theo thứ tự ưu tiên, chứa tất cả các yêu cầu, tính năng và công việc cần thực hiện cho dự án. Nhóm đã xác định các "Epics" (nhóm chức năng lớn) và các "User Stories" (yêu cầu người dùng) tương ứng.



Hình 5. 1 Product Backlog

5.2.2. Kế hoạch Sprint

Dự án được chia thành 4 Sprint chính, mỗi Sprint tập trung vào việc hoàn thành một nhóm các chức năng cụ thể:

Sprint 1: Nền tảng và Chức năng cơ bản

- Mục tiêu: Thiết lập cấu trúc dự án, Docker, CSDL và xây dựng các chức năng hiển thị phim cơ bản.
- Các User Story chính: Xem danh sách phim, xem chi tiết phim.

Sprint 2: Xác thực và Luồng Đặt vé

- Mục tiêu: Xây dựng hoàn chỉnh hệ thống đăng ký, đăng nhập và quy trình đặt vé.
- Các User Story chính: Đăng ký, đăng nhập, chọn rạp, chọn suất chiếu, chọn ghế.

Sprint 3: Hoàn thiện Nghiệp vụ và Chức năng Phụ

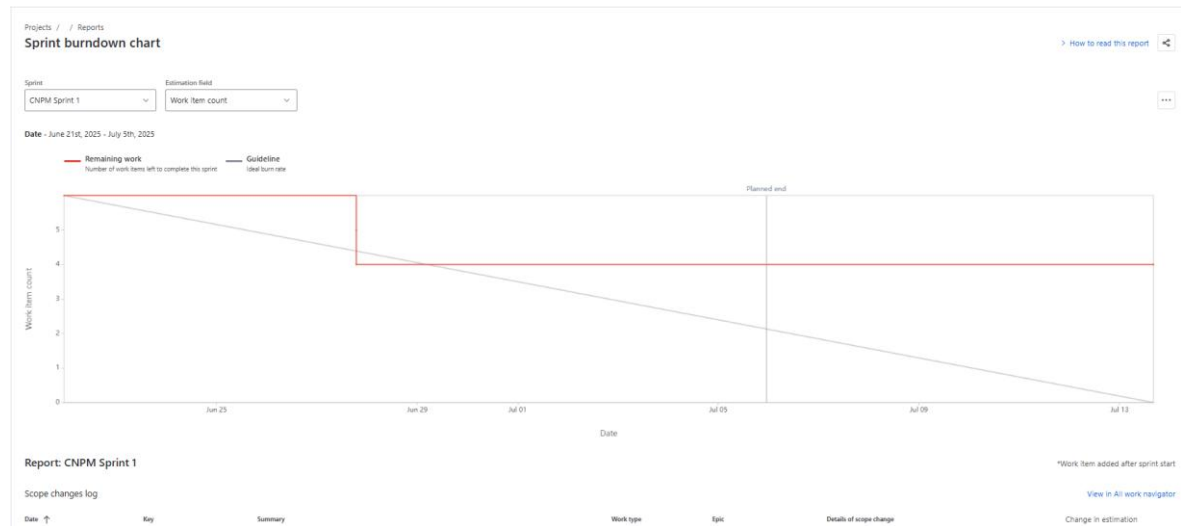
- Mục tiêu: Tích hợp chức năng gửi email, tính giá vé động, và xây dựng các trang tra cứu thông tin.
- Các User Story chính: Gửi email xác nhận, xem lịch sử đặt vé, trang Cụm Rạp, trang Giá Vé.

Sprint 4: Xây dựng Trang Admin và Hoàn thiện

- Mục tiêu: Xây dựng khu vực quản trị cho admin và hoàn thiện các yêu cầu kỹ thuật cuối cùng.
- Các User Story chính: Quản lý Phim/Rạp/Suất chiếu, thiết lập CI/CD, triển khai lên Cloud.

5.3. Biểu đồ Burndown Chart

Burndown Chart là một công cụ trực quan giúp theo dõi tiến độ "đốt cháy" công việc của nhóm so với kế hoạch đã đề ra. Nó giúp nhóm sớm phát hiện các rủi ro về tiến độ và có sự điều chỉnh kịp thời.



Hình 5. 2 Burndown Chart của một Sprint

CHƯƠNG 6: KIỂM THỬ

6.1. Chiến lược kiểm thử

Để đảm bảo chất lượng và sự ổn định của hệ thống CineBooking, nhóm đã áp dụng một chiến lược kiểm thử đa cấp, tập trung vào việc xác minh tính đúng đắn của các chức năng ở cả phía backend và frontend. Chiến lược này bao gồm hai hình thức chính:

Kiểm thử thủ công: Được sử dụng để kiểm tra các luồng nghiệp vụ phức tạp và giao diện người dùng, đảm bảo trải nghiệm người dùng cuối được mượt mà và đúng như thiết kế.

Kiểm thử tự động: Được tích hợp vào quy trình CI/CD để tự động kiểm tra chất lượng mã nguồn mỗi khi có thay đổi, giúp phát hiện lỗi và giảm thiểu rủi ro.

6.2. Công cụ kiểm thử

Postman: Được sử dụng làm công cụ chính để thực hiện kiểm thử thủ công cho các API của backend. Postman cho phép gửi các request HTTP tùy chỉnh đến các endpoint, kiểm tra các kịch bản thành công và thất bại, và xác minh tính đúng đắn của dữ liệu trả về.

Jest: Framework kiểm thử được tích hợp sẵn trong NestJS, được sử dụng để viết các bài kiểm thử đơn vị (unit test) cho các service và controller ở backend.

ESLint: Công cụ phân tích mã tĩnh, được sử dụng để tự động kiểm tra và đảm bảo mã nguồn tuân thủ các quy tắc về chất lượng và cú pháp.

GitHub Actions: Nền tảng CI/CD được sử dụng để tự động hóa toàn bộ quy trình kiểm thử.

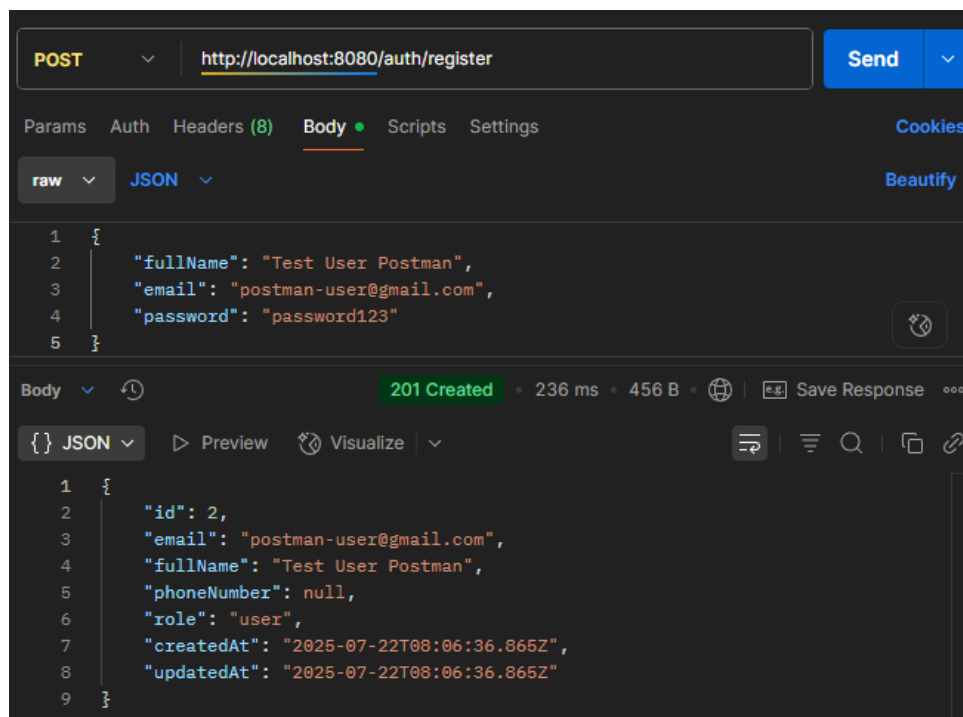
6.3. Kết quả kiểm thử API với Postman

Nhóm đã sử dụng Postman để kiểm thử toàn bộ các API của hệ thống, tập trung vào các luồng nghiệp vụ quan trọng. Các kịch bản kiểm thử bao gồm:

Luồng Xác thực: Kiểm tra thành công các API đăng ký, đăng nhập và xử lý các trường hợp lỗi như email trùng lặp hoặc sai mật khẩu.

Luồng Đặt vé: Kiểm tra API đặt vé với token xác thực hợp lệ, xử lý các trường hợp đặt ghế đã có người chọn, và xác minh rằng email xác nhận được gửi đi thành công.

Luồng Quản trị viên: Kiểm tra các API CRUD (Thêm, Sửa, Xóa) cho Phim, Rạp, Suất chiếu, đảm bảo chúng được bảo vệ và chỉ có tài khoản admin mới có thể truy cập.



Hình 6. 1 request đặt vé thành công trên Postman

6.4. Kiểm thử tự động với GitHub Actions

Quy trình CI/CD được thiết lập với GitHub Actions đóng vai trò là hàng rào bảo vệ chất lượng cuối cùng. Mỗi khi có một commit mới được đẩy lên repository, hai workflow (một cho frontend, một cho backend) sẽ được tự động kích hoạt.

Đối với Backend:

Kiểm tra Linting (`npm run lint`): Tự động rà soát toàn bộ mã nguồn TypeScript để phát hiện các lỗi cú pháp, vi phạm quy tắc code, và các vấn đề về bảo mật tiềm ẩn (như sử dụng kiểu any không an toàn).

Kiểm tra Build (`npm run build`): Biên dịch toàn bộ dự án từ TypeScript sang JavaScript để đảm bảo không có lỗi về kiểu dữ liệu hay các lỗi biên dịch khác.

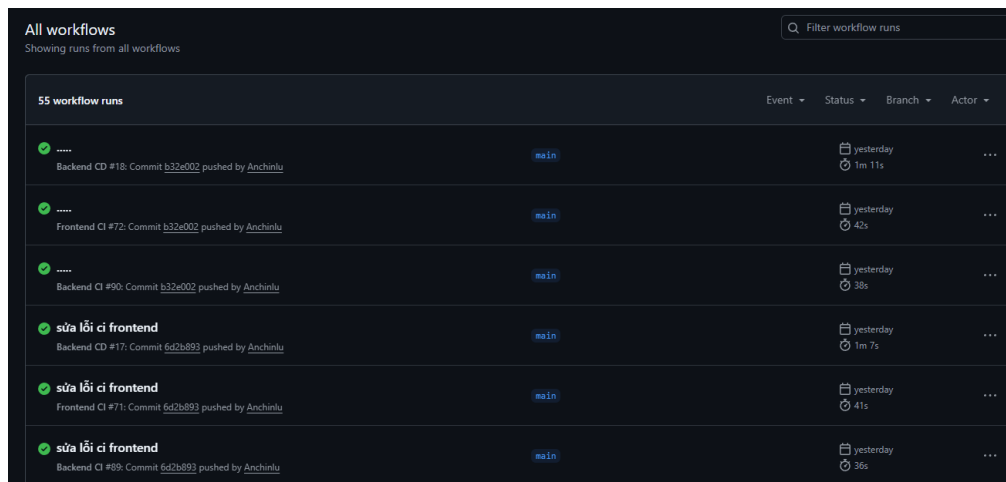
Kiểm tra Unit Test (`npm run test`): Tự động chạy các bài kiểm thử đơn vị đã được viết bằng Jest để xác minh rằng các thành phần logic cốt lõi (ví dụ: các hàm trong service) vẫn hoạt động đúng như mong đợi sau khi có thay đổi.

Đối với Frontend:

Kiểm tra Linting (`npm run lint`): Tương tự như backend, đảm bảo chất lượng code React/TypeScript.

Kiểm tra Build (`npm run build`): Chạy next build để tạo một phiên bản production của ứng dụng, giúp phát hiện sớm các lỗi về kiểu dữ liệu hoặc các vấn đề tương thích trong các component.

Quy trình này đảm bảo rằng mọi thay đổi được đưa vào nhánh chính đều đã vượt qua các bài kiểm tra chất lượng cơ bản, giúp giảm thiểu đáng kể rủi ro gây lỗi cho sản phẩm.



The screenshot displays the 'All workflows' page in GitHub Actions. It shows a list of 55 workflow runs. The table includes columns for Event, Status, Branch, and Actor. The runs are categorized by workflow name and commit hash. The first three runs are for 'Backend CD' and 'Frontend CI' workflows, all with a status of 'Success'. The next three runs are for 'sửa lỗi ci frontend' workflows, also with a status of 'Success'. The runs are sorted by time, with the most recent at the top.

Event	Status	Branch	Actor
Push	Success	main	anchinlu
Push	Success	main	anchinlu
Push	Success	main	anchinlu
Push	Success	main	anchinlu
Push	Success	main	anchinlu
Push	Success	main	anchinlu

Hình 6. 2 GitHub Actions

CHƯƠNG 7: ĐÁNH GIÁ VÀ KẾT LUẬN

7.1. Đánh giá kết quả đạt được

Sau quá trình thực hiện, dự án "Hệ thống Đặt vé Xem phim Trực tuyến - CineBooking" đã hoàn thành các mục tiêu chính đã đề ra. Nhóm đã xây dựng thành công một ứng dụng web hoàn chỉnh, đáp ứng đầy đủ các yêu cầu chức năng cốt lõi và áp dụng thành thạo các công nghệ, quy trình phát triển phần mềm hiện đại.

Về mặt chức năng: Hệ thống đã cung cấp một luồng nghiệp vụ trọn vẹn cho cả người dùng và quản trị viên. Người dùng có thể dễ dàng tra cứu thông tin, thực hiện đặt vé và nhận xác nhận qua email. Quản trị viên có một khu vực riêng để quản lý các tài nguyên quan trọng của hệ thống.

Về mặt kỹ thuật: Dự án đã tuân thủ nghiêm ngặt kiến trúc Client-Server, sử dụng RESTful API, và được đóng gói hoàn toàn bằng Docker. Đặc biệt, việc tích hợp thành công quy trình CI/CD với GitHub Actions đã giúp tự động hóa việc kiểm tra chất lượng mã nguồn, đảm bảo tính ổn định của sản phẩm.

Về mặt quy trình: Nhóm đã vận dụng mô hình Scrum và công cụ Jira để quản lý dự án một cách có hệ thống, đảm bảo tiến độ và sự phối hợp hiệu quả giữa các thành viên.

7.2. Những khó khăn gặp phải trong quá trình thực hiện

Trong quá trình xây dựng dự án, nhóm đã đối mặt với một số khó khăn và thách thức đáng kể:

Lỗi tương thích môi trường: Việc thiết lập và đồng bộ môi trường giữa máy phát triển (local) và môi trường triển khai (production) ban đầu gặp nhiều khó khăn, đặc biệt là các vấn đề liên quan đến kết nối mạng trong Docker và cấu hình biến môi trường giữa Vercel và Render.

Xử lý lỗi bất đồng bộ: Việc gỡ lỗi các vấn đề liên quan đến Promise và các thao tác bất đồng bộ trong Next.js (đặc biệt là với Server Components) đòi hỏi nhiều thời gian nghiên cứu và thử nghiệm.

Tích hợp CI/CD: Việc thiết lập các file workflow cho GitHub Actions lần đầu tiên gặp nhiều lỗi liên quan đến cú pháp và các quy tắc lint nghiêm ngặt, đòi hỏi nhóm phải kiên nhẫn sửa lỗi và tối ưu hóa mã nguồn.

Quản lý CSDL: Việc sao chép và đồng bộ dữ liệu (đặc biệt là dữ liệu có Tiếng Việt) giữa môi trường local và cloud (Render) gặp phải các vấn đề về mã hóa ký tự và xung đột khi tạo bảng.

7.3. Bài học rút ra và đề xuất cải thiện trong tương lai

Qua những thách thức trên, nhóm đã rút ra được nhiều bài học kinh nghiệm quý báu:

Tầm quan trọng của CI/CD: Việc thiết lập quy trình kiểm thử tự động ngay từ đầu giúp phát hiện lỗi sớm, đảm bảo chất lượng code và tiết kiệm rất nhiều thời gian gỡ lỗi ở các giai đoạn sau.

Sự cần thiết của việc viết code "sạch": Tuân thủ các quy tắc linting và viết code một cách tường minh giúp giảm thiểu lỗi và làm cho dự án dễ bảo trì hơn.

Kỹ năng gỡ lỗi hệ thống phân tán: Hiểu rõ cách các thành phần (Frontend, Backend, Database) giao tiếp với nhau trong các môi trường khác nhau là chìa khóa để giải quyết các vấn đề phức tạp.

Đề xuất cải thiện và hướng phát triển trong tương lai:

Tích hợp cổng thanh toán trực tuyến: Tích hợp các cổng thanh toán như Momo, ZaloPay, hoặc VNPAY để hoàn thiện luồng đặt vé.

Nâng cao trải nghiệm người dùng: Xây dựng các tính năng như bình luận, đánh giá phim, gợi ý phim dựa trên lịch sử xem.

Tối ưu hóa hiệu năng: Áp dụng các kỹ thuật caching nâng cao ở cả backend và frontend để tăng tốc độ tải trang và phản hồi của API.

Mở rộng hệ thống quản trị: Bổ sung các biểu đồ và báo cáo chi tiết hơn trong trang admin để giúp quản lý doanh thu và phân tích hành vi người dùng.

CHƯƠNG 8: PHỤ LỤC

8.1. Hướng dẫn cài đặt và chạy ứng dụng

Để cài đặt và chạy dự án CineBooking trên máy tính cá nhân, cần đảm bảo các công cụ sau đã được cài đặt:

Docker Desktop: Công cụ để quản lý và chạy các container.

Git: Hệ thống quản lý phiên bản phân tán.

Node.js và npm: Môi trường chạy và quản lý các gói phụ thuộc cho JavaScript.

Các bước thực hiện:

Sao chép mã nguồn: Mở terminal và chạy lệnh sau để sao chép (clone) repository của dự án về máy:

```
git clone https://github.com/Anchinlu/Cong_nghe_phan_mem.git
```

Cấu hình biến môi trường:

Di chuyển vào thư mục backend của dự án.

Tạo một file mới tên là .env.

Sao chép toàn bộ nội dung từ file .env.example (nếu có) hoặc điền các thông tin cần thiết như sau:

```
# Cấu hình Cơ sở dữ liệu
DATABASE_HOST=db
DATABASE_PORT=5432
DATABASE_USERNAME=your_db_user
DATABASE_PASSWORD=your_db_password
DATABASE_NAME=your_db_name

# Khóa bí mật cho JWT
JWT_SECRET=your_super_secret_key

# Cấu hình gửi Email
MAIL_HOST=smtp.gmail.com
MAIL_USER=your_email@gmail.com
MAIL_PASSWORD=your_google_app_password
```

Khởi chạy hệ thống:

Quay lại thư mục gốc của dự án.

Mở terminal và chạy lệnh sau:

```
docker-compose up --build
```

Lệnh này sẽ tự động build các image cho frontend và backend, sau đó khởi tạo toàn bộ 3 container (database, backend, frontend).

Truy cập ứng dụng:

Frontend (Trang web): Mở trình duyệt và truy cập địa chỉ <http://localhost:3000>.

Backend (API Docs): Truy cập <http://localhost:8080/api-docs> để xem tài liệu API Swagger.

8.2. Liên kết GitHub Repository

Toàn bộ mã nguồn của dự án được lưu trữ và quản lý tại repository sau:

URL: https://github.com/Anchinlu/Cong_nghe_phan_mem

8.3. Liên kết Demo sản phẩm

Ứng dụng đã được triển khai và có thể truy cập trực tuyến tại các địa chỉ sau:

Frontend (Vercel): <https://cong-nghe-phan-mem.vercel.app/>

Backend (Render): <https://cinebooking-backend.onrender.com>