FINAL PROJECT REPORT

SEMESTER 3, ACADEMIC YEAR: 2024-2025

CT313H: WEB TECHNOLOGIES AND SERVICES

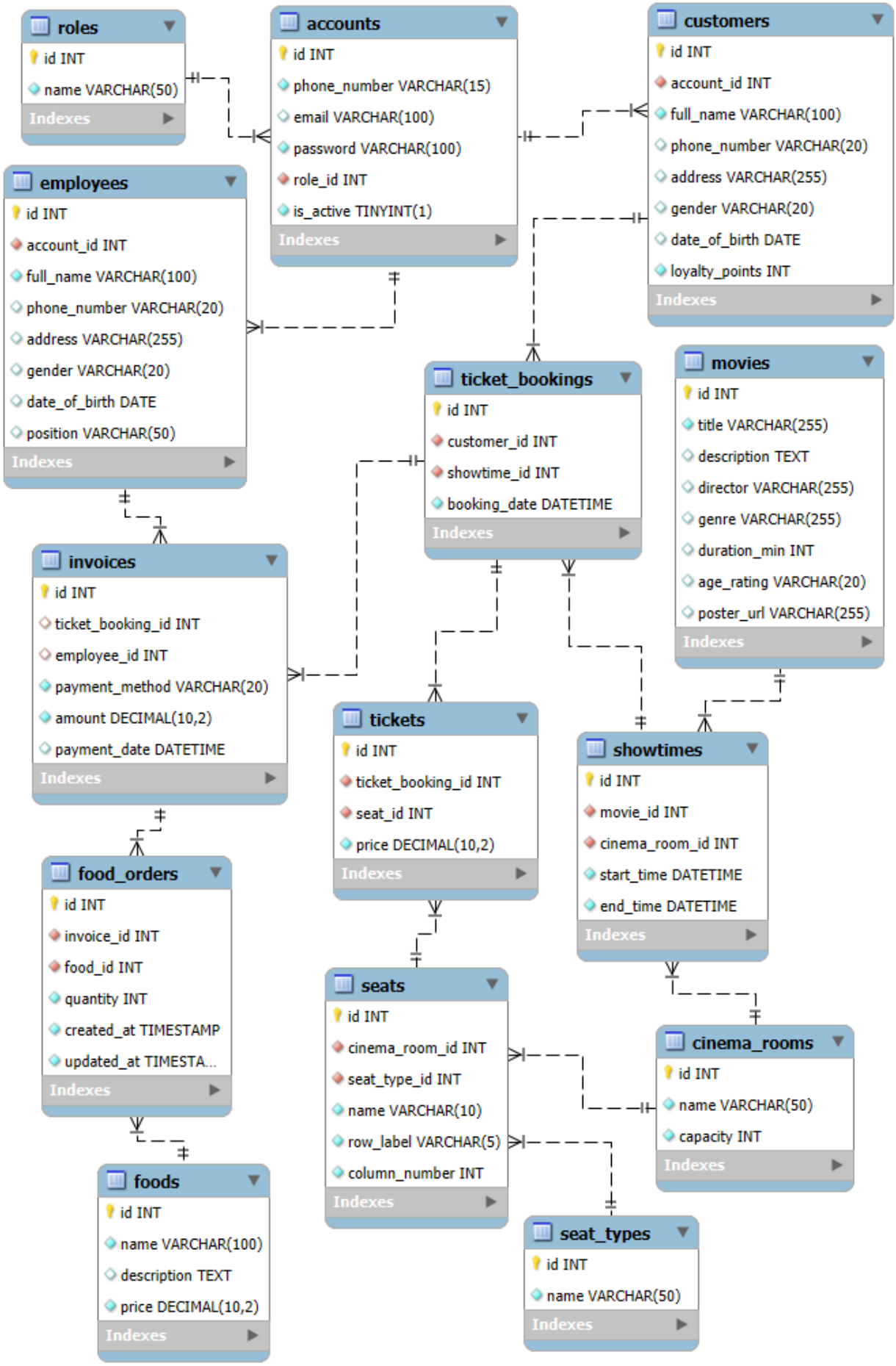
- **Project/Application name: Cinema**
- **GitHub links (for both frontend and backend):**

https://github.com/24-25Sem3-Courses/ct313hm02-project-AndrewNguyenITVN.git

- **Student ID 1: B2203438**
- **Student Name 1: Huỳnh Tấn Đạt**
- **Student ID 2: B2205896**
- **Student Name 2: Nguyễn Minh Nhựt**
- **Youtube link:** https://youtu.be/WFCSNUXTSU0?si=rtq8mqfrp1SjBgKY
- **Class/Group Number (e.g, CT313HM01): CT313HM02**

I. Introduction

- Project/application description: A single page application (SPA) developed to comprehensively manage a cinema system. The system provides a full-featured solution for managing and operating a cinema, including functionalities such as movie management, employees management, customers management, online ticket booking, and food & beverage service.
- A list of tables and their structures in the database (could show a CDM/PDM diagram here).


roles
id INT
name VARCHAR(50)
Indexes
accounts
id INT
phone_number VARCHAR(15)
email VARCHAR(100)
password VARCHAR(100)
role_id INT
is_active TINYINT(1)
Indexes
customers
id INT
account_id INT
full_name VARCHAR(100)
phone_number VARCHAR(20)
address VARCHAR(255)
gender VARCHAR(20)
date_of_birth DATE
loyalty_points INT
Indexes
employees
id INT
account_id INT
full_name VARCHAR(100)
phone_number VARCHAR(20)
address VARCHAR(255)
gender VARCHAR(20)
date_of_birth DATE
position VARCHAR(50)
Indexes
ticket_bookings
id INT
customer_id INT
showtime_id INT
booking_date DATETIME
Indexes
movies
id INT
title VARCHAR(255)
description TEXT
director VARCHAR(255)
genre VARCHAR(255)
duration_min INT
age_rating VARCHAR(20)
poster_url VARCHAR(255)
Indexes
invoices
id INT
ticket_booking_id INT
employee_id INT
payment_method VARCHAR(20)
amount DECIMAL(10,2)
payment_date DATETIME
Indexes
tickets
id INT
ticket_booking_id INT
seat_id INT
price DECIMAL(10,2)
Indexes
showtimes
id INT
movie_id INT
cinema_room_id INT
start_time DATETIME
end_time DATETIME
Indexes
food_orders
id INT
invoice_id INT
food_id INT
quantity INT
created_at TIMESTAMP
updated_at TIMESTA...
Indexes
seats
id INT
cinema_room_id INT
seat_type_id INT
name VARCHAR(10)
row_label VARCHAR(5)
column_number INT
Indexes
cinema_rooms
id INT
name VARCHAR(50)
capacity INT
Indexes
foods
id INT
name VARCHAR(100)
description TEXT
price DECIMAL(10,2)
Indexes
seat_types
id INT
name VARCHAR(50)
Indexes

- A task assignment sheet for each member if working in groups.

No.	Task Description	Member
1	Organize structure, design database, initialize the API server.	Nguyen Minh Nhut
2	Implement REST API for CRUD operations and document them with OpenAPI (Swagger)	Nguyen Minh Nhut Huynh Tan Dat
3	Movies, showtimes management, Cinema rooms, food management API	Huynh Tan Dat
4	Authentication system, booking system management API	Nguyen Minh Nhut
5	Implement client-side state management using TanStack Query and Pinia	Huynh Tan Dat Nguyen Minh Nhut
6	Design responsive UI for Vue components with Bootstrap: AuthForm, ChangePasswordForm, EmployeeForm, CustomerForm, MovieCard, UserCard, MainPagination, InputSearch,...	Huynh Tan Dat Nguyen Minh Nhut
7	Implement Vue pages: LoginPage, HomePage, ProfilePage, MovieAddPage, MovieDetailPage, BookingPage, MyBookingPage, CheckBooking, EmployeeAdd, EmployeeEdit, CustomerList, CustomerEdit, NotFound, ForbiddenPage,...	Nguyen Minh Nhut Huynh Tan Dat
8	Rate limiting backend	Huynh Tan Dat
9	Build and deploy the SPA with Caddy, deploy the backend API with PM2	Nguyen Minh Nhut

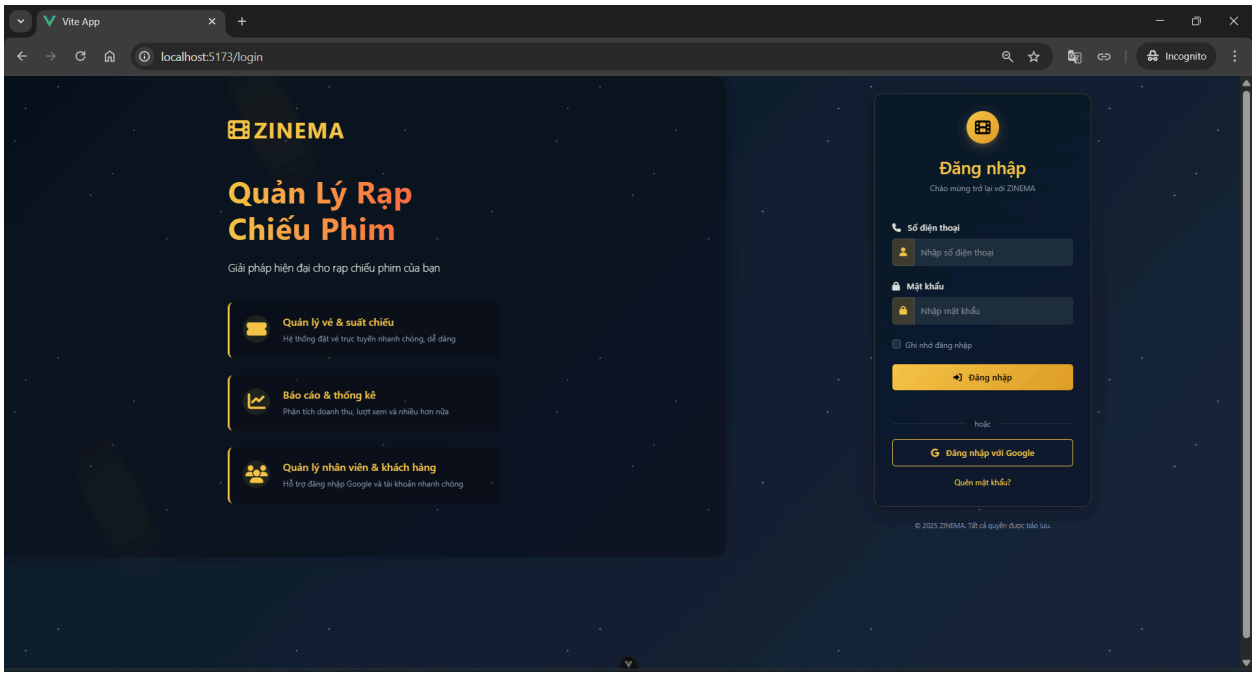
II. Details of implemented features

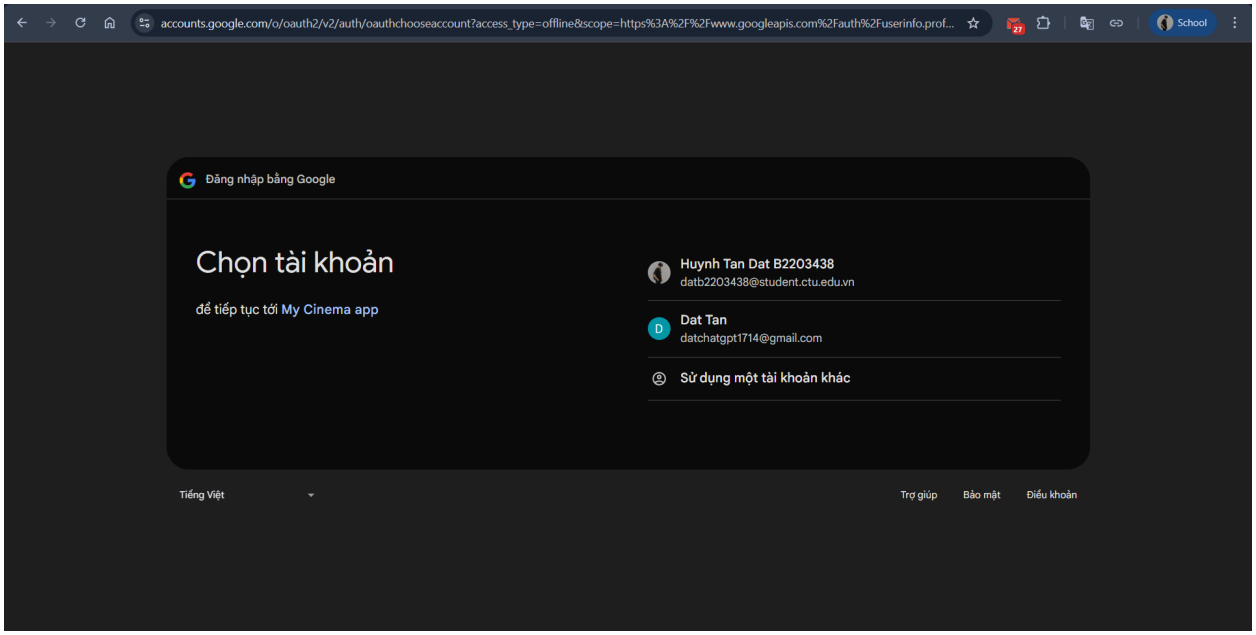
1. Authentication & User Management

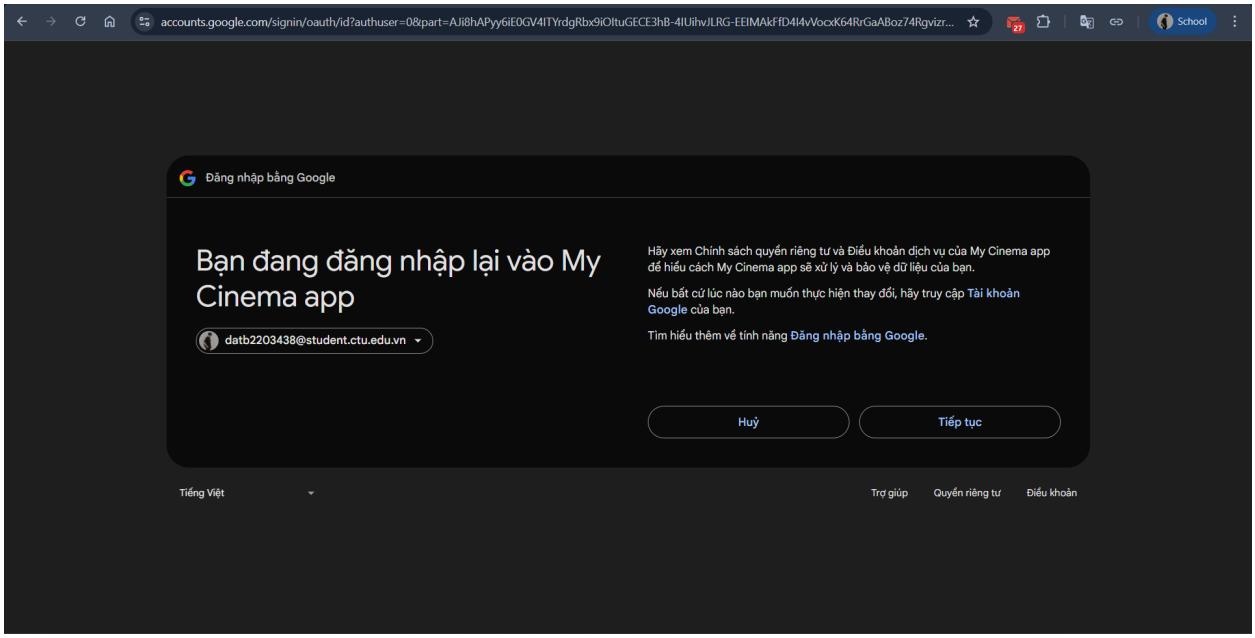
- Description: This feature manages user authentication and user management for the cinema system. It provides login functionality using both local accounts (phone number + password) and Google OAuth integration. The system supports three user roles: admin, staff, and customer. Users can view and update

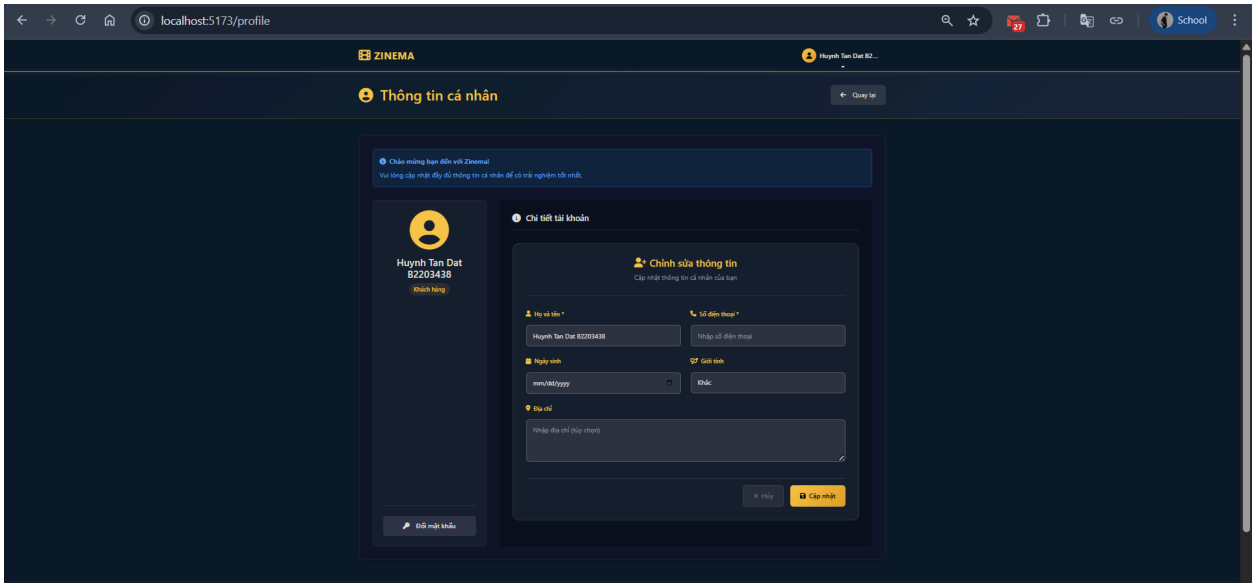
their personal information, while admins can manage employee accounts and view customer information.

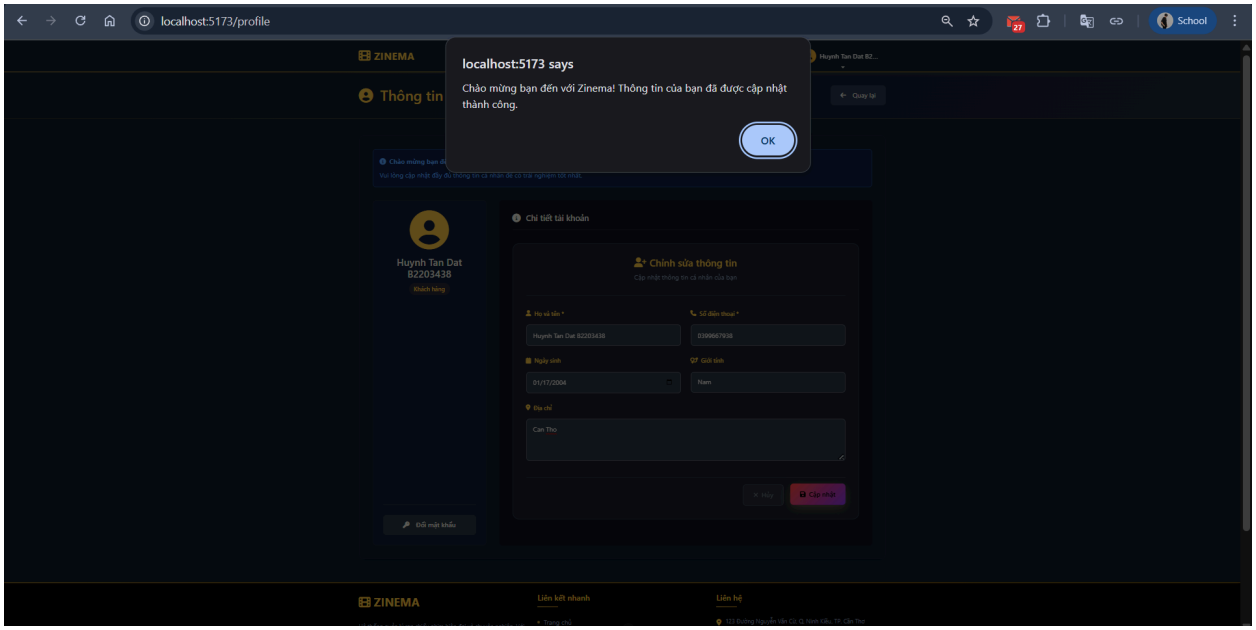
- Screenshots:

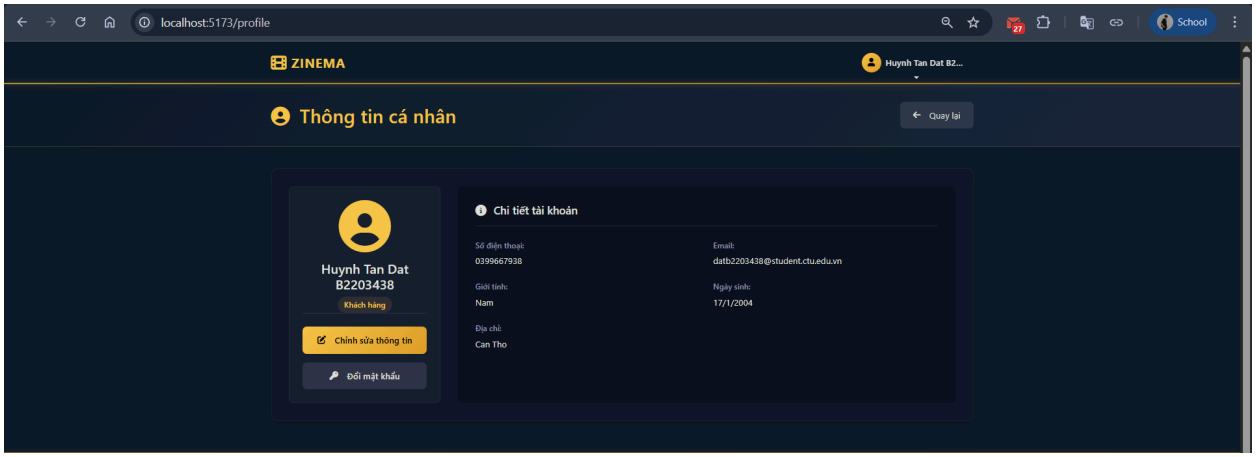





accounts.google.com/signin/oauth/id?authuser=0&part=AJi8hAPyy6iE0GV4ITYrdgRbx9iOltuGECE3hB-4IUihvJLRG-EEIMAkFfD4l4vVocxK64RrGaABoz74Rgvizr...
School
Đăng nhập bằng Google
Bạn đang đăng nhập lại vào My Cinema app
datb2203438@student.ctu.edu.vn
Hãy xem Chính sách quyền riêng tư và Điều khoản dịch vụ của My Cinema app để hiểu cách My Cinema app sẽ xử lý và bảo vệ dữ liệu của bạn.
Nếu bất cứ lúc nào bạn muốn thực hiện thay đổi, hãy truy cập Tài khoản Google của bạn.
Tìm hiểu thêm về tính năng Đăng nhập bằng Google.
Huỷ
Tiếp tục
Tiếng Việt
Trợ giúp
Quyền riêng tư
Điều khoản


localhost:5173/profile
School
ZINEMA
Thông tin cá nhân
Quay lại
Chào mừng bạn đến với Zinema!
Vui lòng cập nhật đầy đủ thông tin cá nhân để có trải nghiệm tốt nhất.
Huynh Tan Dat B2203438
Khách hàng
Đổi mật khẩu
Chi tiết tài khoản
Chỉnh sửa thông tin
Cập nhật thông tin cá nhân của bạn
Họ và tên *
Huynh Tan Dat B2203438
Số điện thoại *
Nhập số điện thoại
Ngày sinh
mm/dd/yyyy
Giới tính
Khác
Địa chỉ
Nhập địa chỉ (tùy chọn)
Hủy
Cập nhật





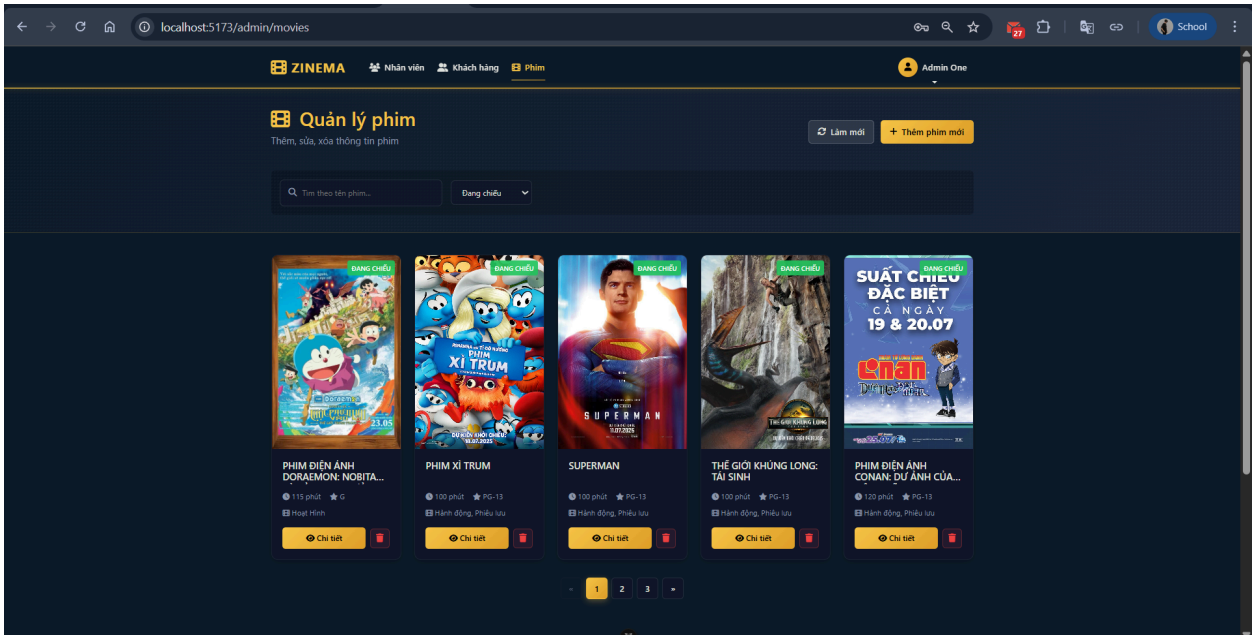
- Implementation details:
  - This feature uses jsonwebtoken on the backend to create and verify JWTs for session management. bcryptjs is used for securely hashing user passwords. jwt-decode extracts info from the token, and pinia manages global user state.
  - Which server-side APIs does this feature use:
    - User Login (Local Account):
      - Endpoint: POST /api/auth/login
      - Data sent: The request body contains user credentials including phone_number, and password. The data is sent in JSON format.

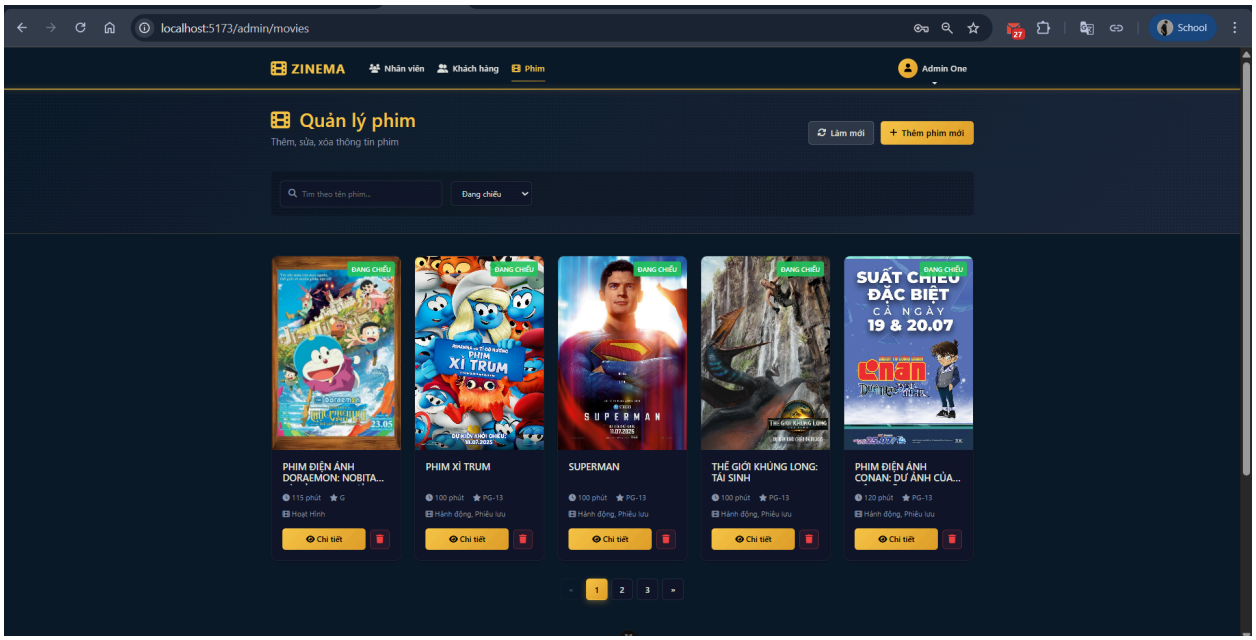
        - Response: A JSON object containing a status field indicating success, and a data object with a token field containing the JWT authentication token.
    - Google OAuth Initiation:
        - Endpoint: GET /api/auth/google
        - Data sent: No request body required.
        - Response: Server redirects the user to the Google OAuth consent screen for authentication.
    - Google OAuth Callback:
        - Endpoint: GET /api/auth/google/callback
        - Data sent: Google provides a code in the query string parameter.
        - Response: A JSON object containing the JWT token for the authenticated user.
    - Update Customer Profile:
        - Endpoint: PUT /api/auth/customers/:id
        - Data sent: Customer ID in the URL path parameter, a JWT token in the Authorization header, and a request body in JSON format containing the fields to be updated (e.g., full_name, email, date_of_birth).
        - Response: A JSON object containing the complete and updated customer profile information.
- Does this feature read/store data? In which table?
    - It reads from the roles table to verify user permissions and role information.
    - It reads from and writes to the accounts table for storing login credentials and role associations.
    - The feature reads from and writes to the customers table for storing customer profile information.
    - It also reads from and writes to the employees table for storing employee profile information.
- Which client-side states are needed to implement this feature?
    - user: A Pinia-stored object with the logged-in user's info, persisted in localStorage.
    - token: The raw JWT string, persisted in localStorage.
    - isAuthenticated: A computed boolean in Pinia derived from the user and token state.

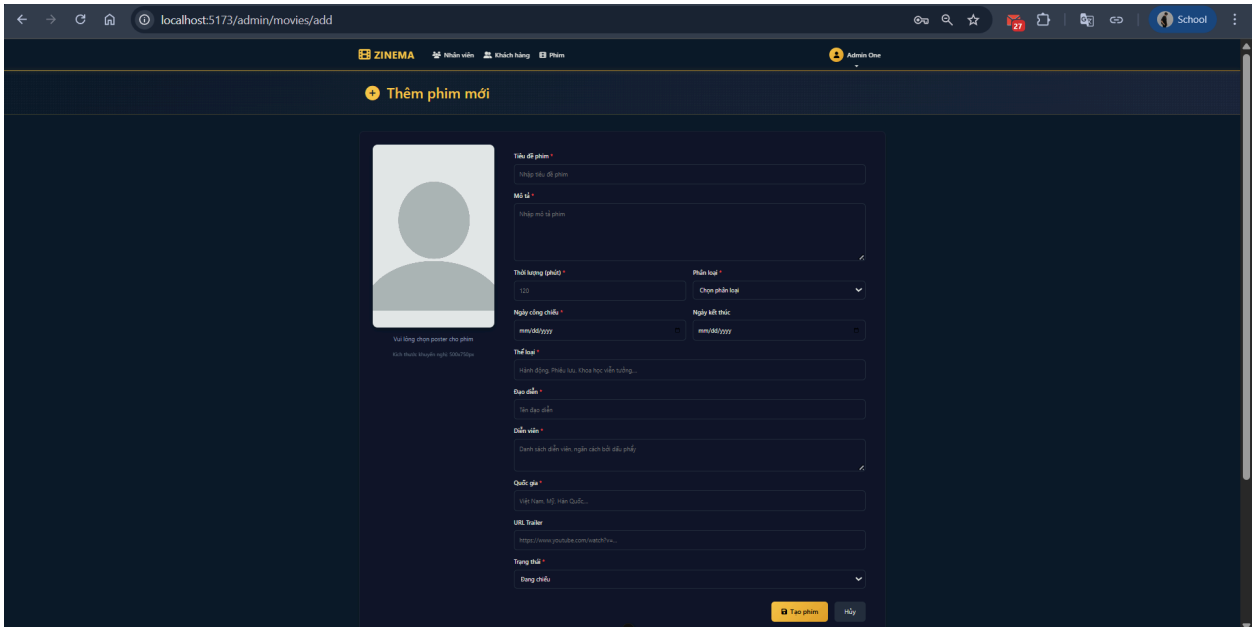
- Component-local state for form inputs and loading/error indicators.

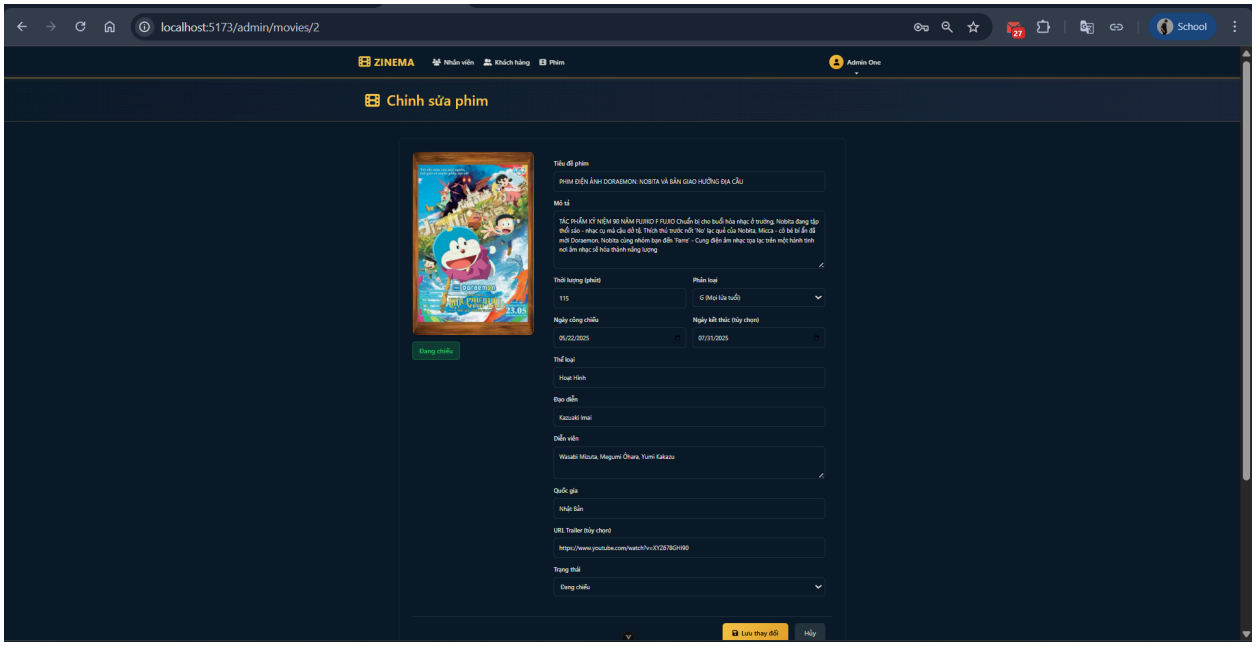
2. Movie Management

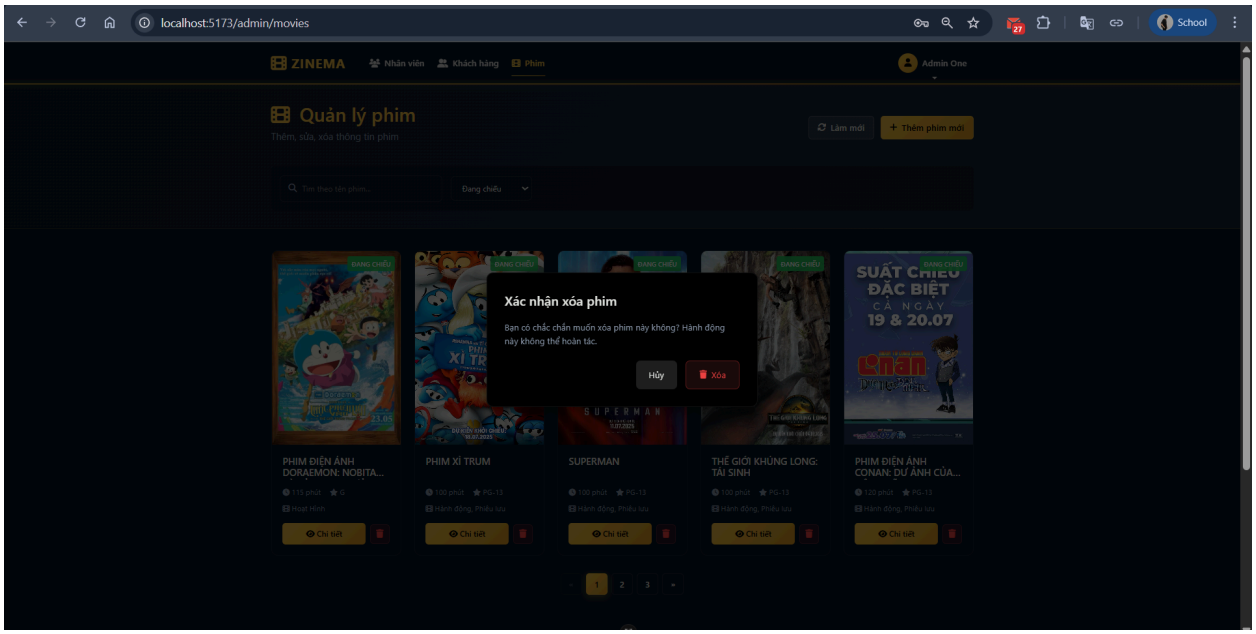
- Description: This feature handles the complete movie management for the cinema system. It allows all users to browse and search for movies, while providing admin users the ability to add, update, and remove movies.
- Screenshots:




localhost:5173/admin/movies
ZINEMA
Nhân viên
Khách hàng
Phim
Admin One
Quản lý phim
Thêm, sửa, xóa thông tin phim
Làm mới
+ Thêm phim mới
Tìm theo tên phim...
Đang chiếu
ĐANG CHIẾU
PHIM ĐIỆN ẢNH DORAEMON: NOBITA...
PHIM XÌ TRUM
SUPERMAN
THẾ GIỚI KHỦNG LONG: TÁI SINH
PHIM ĐIỆN ẢNH CONAN: DƯ ẢNH CỦA...
SUẤT CHIẾU ĐẶC BIỆT
CẢ NGÀY
19 & 20.07
115 phút
100 phút
120 phút
PG-13
G
Hoạt Hình
Hành động, Phiêu lưu
Chi tiết
1
2
3


localhost:5173/admin/movies/add
ZINEMA
Nhân viên
Khách hàng
Phim
Admin One
Thêm phim mới
Tiêu đề phim *
Nhập tiêu đề phim
Mô tả *
Nhập mô tả phim
Thời lượng (phút) *
120
Phân loại *
Chọn phân loại
Ngày công chiếu *
mm/dd/yyyy
Ngày kết thúc
mm/dd/yyyy
Thể loại *
Hành động, Phiêu lưu, Khoa học viễn tưởng...
Đạo diễn *
Tên đạo diễn
Diễn viên *
Danh sách diễn viên, ngăn cách bởi dấu phẩy
Quốc gia *
Việt Nam, Mỹ, Hàn Quốc...
URL Trailer
https://www.youtube.com/watch?v=...
Trạng thái *
Đang chiếu
Tạo phim
Hủy
Vui lòng chọn poster cho phim





- Implementation details:
  - The backend uses multer for multipart/form-data uploads (movie posters). The frontend uses Vue's local component state for forms and the custom `efetch` function for API calls.

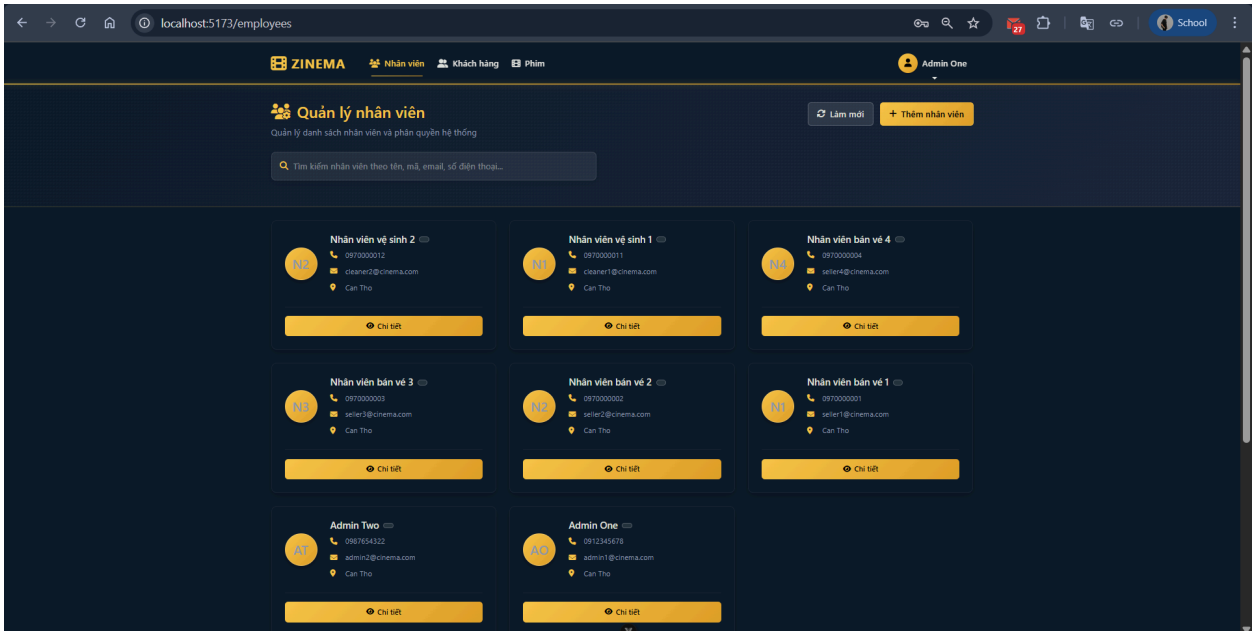
  - Which server-side APIs does this feature use:
    - Get All Movies:
      - Endpoint: GET /api/movies

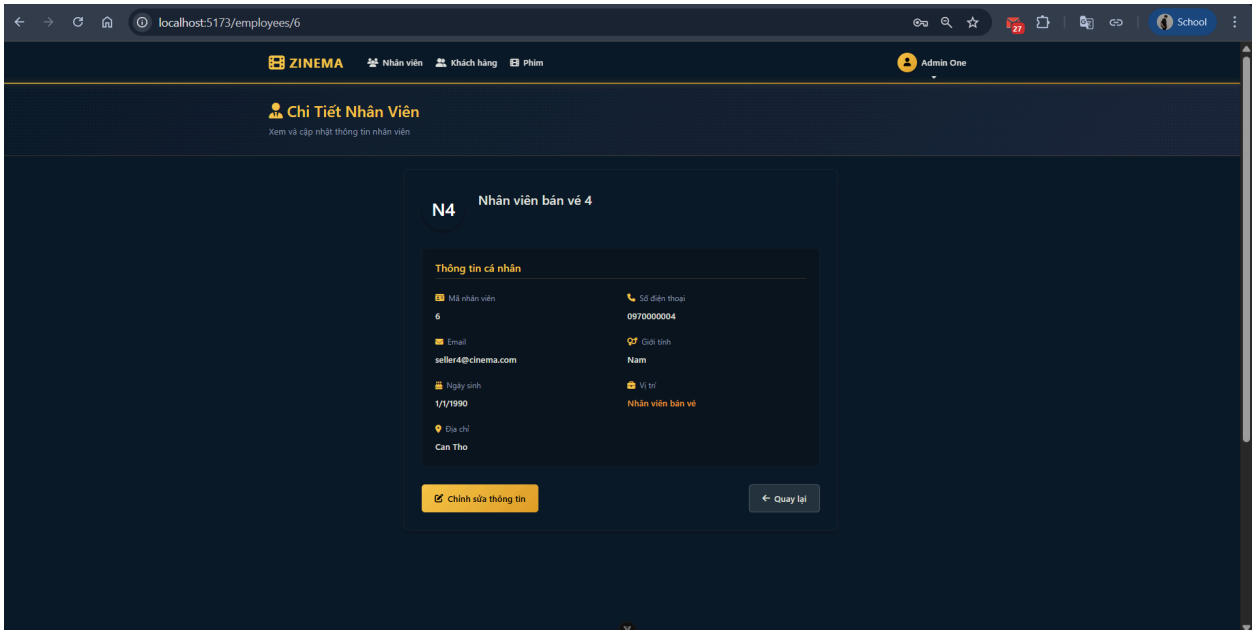
        - Data sent: Optional query parameters including title for searching, page for pagination, and limit for results per page.
        - Response: A JSON object containing a movies array and pagination metadata.
    - Get Movie by ID:
        - Endpoint: GET /api/movies/:id
        - Data sent: Movie ID in the URL path parameter.
        - Response: A JSON object containing the complete information for a single movie.
    - Create New Movie (Admin only):
        - Endpoint: POST /api/movies
        - Data sent: JWT token with admin role in Authorization header, and multipart/form-data containing movie details (e.g., title, director, release_date) and the poster image file.
        - Response: A JSON object containing the newly created movie's information.
    - Update Movie (Admin only):
        - Endpoint: PUT /api/movies/:id
        - Data sent: Movie ID in URL path parameter, JWT token with admin role in Authorization header, and multipart/form-data with the fields to be updated.
        - Response: A JSON object containing the updated movie's information.
    - Delete Movie (Admin only):
        - Endpoint: DELETE /api/movies/:id
        - Data sent: Movie ID in the URL path parameter and a JWT token with admin role in the Authorization header.
        - Response: A JSON success message confirming the movie deletion.
- Does this feature read/store data? In which table?
    - It reads from and writes to the movies table. It reads records for displaying movie lists and details. It writes new records when creating movies and updates existing records when modifying them. The delete operation is a soft delete, updating a status or flag to maintain data integrity with related showtimes and bookings.
- Which client-side states are needed to implement this feature?

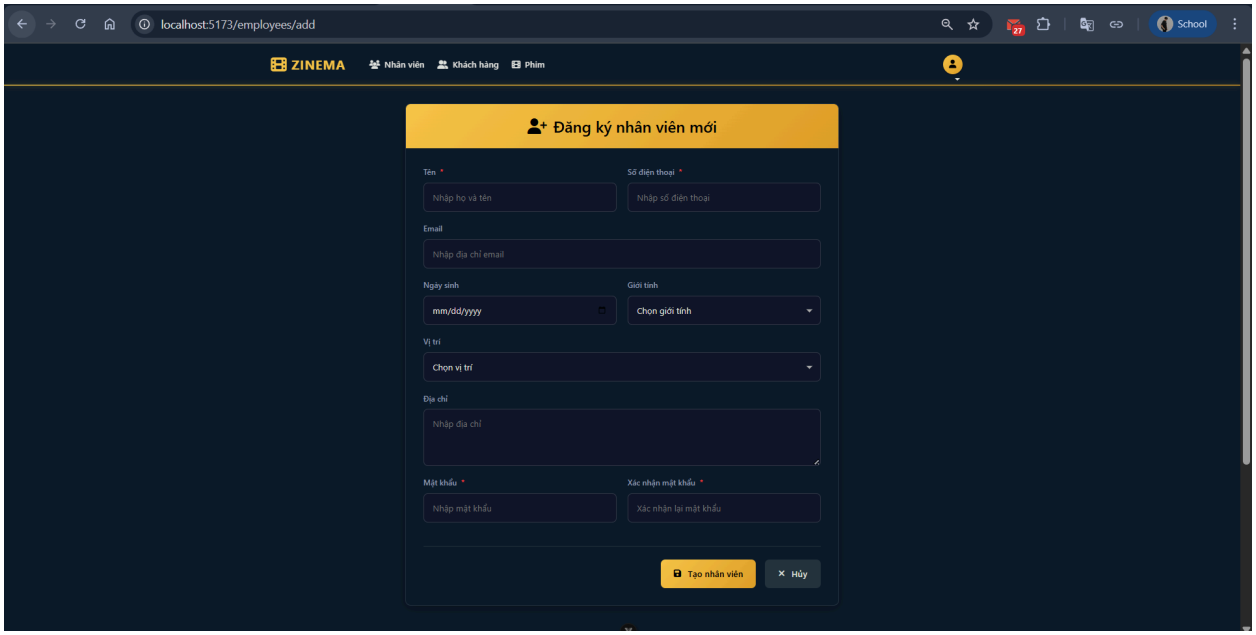
- movies: An array holding the list of movies.
- movie: An object for the "Add"/"Edit" form.
- pagination: An object for page state (currentPage, totalPages).
- isLoading, error: Local state for API call feedback.

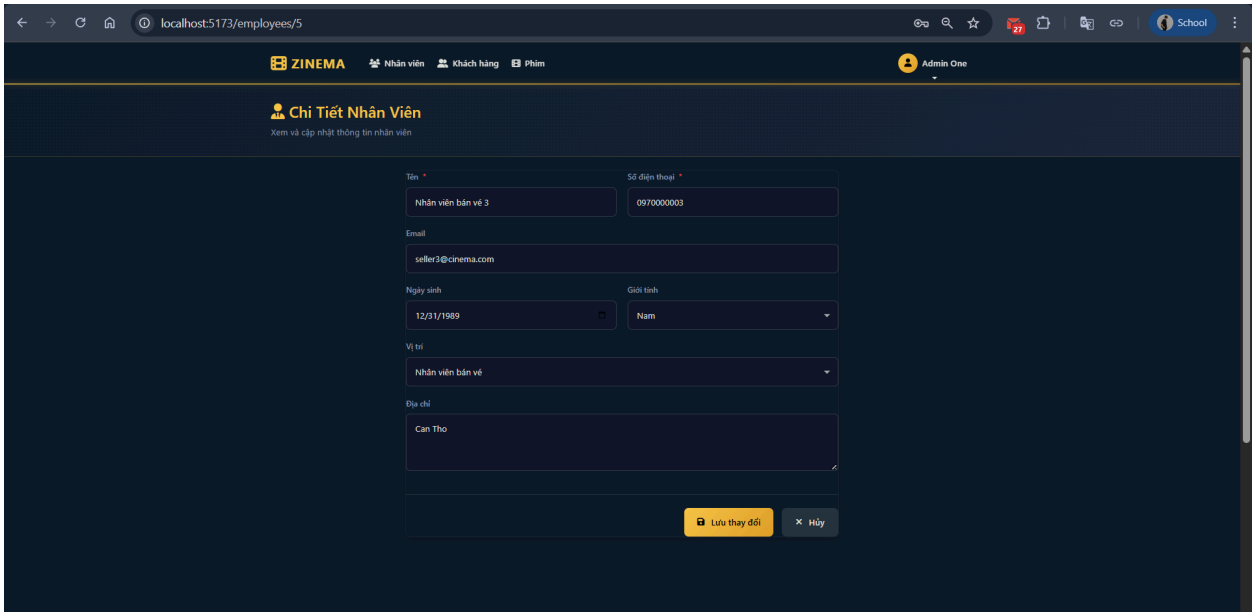
3. Employee Management

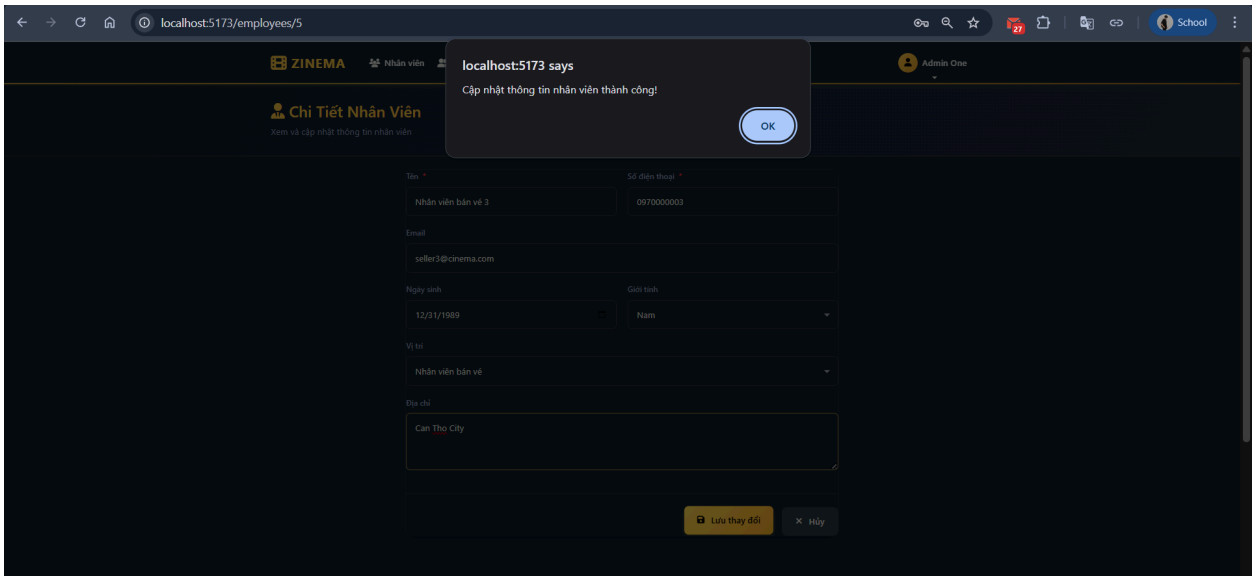
- Description: This feature allows administrators to manage employee accounts. Admins can view a list of all employees, create new employee accounts, update existing employee information, and delete employee accounts.
- Screenshots:




localhost:5173/employees/6
ZINEMA
Nhân viên
Khách hàng
Phim
Admin One
Chi Tiết Nhân Viên
Xem và cập nhật thông tin nhân viên
N4
Nhân viên bán vé 4
Thông tin cá nhân
Mã nhân viên
6
Số điện thoại
0970000004
Email
seller4@cinema.com
Giới tính
Nam
Ngày sinh
1/1/1990
Vị trí
Nhân viên bán vé
Địa chỉ
Can Tho
Chỉnh sửa thông tin
Quay lại


localhost:5173/employees/add
ZINEMA
Nhân viên
Khách hàng
Phim
Đăng ký nhân viên mới
Tên *
Nhập họ và tên
Số điện thoại *
Nhập số điện thoại
Email
Nhập địa chỉ email
Ngày sinh
mm/dd/yyyy
Giới tính
Chọn giới tính
Vị trí
Chọn vị trí
Địa chỉ
Nhập địa chỉ
Mật khẩu *
Nhập mật khẩu
Xác nhận mật khẩu *
Xác nhận lại mật khẩu
Tạo nhân viên
Hủy



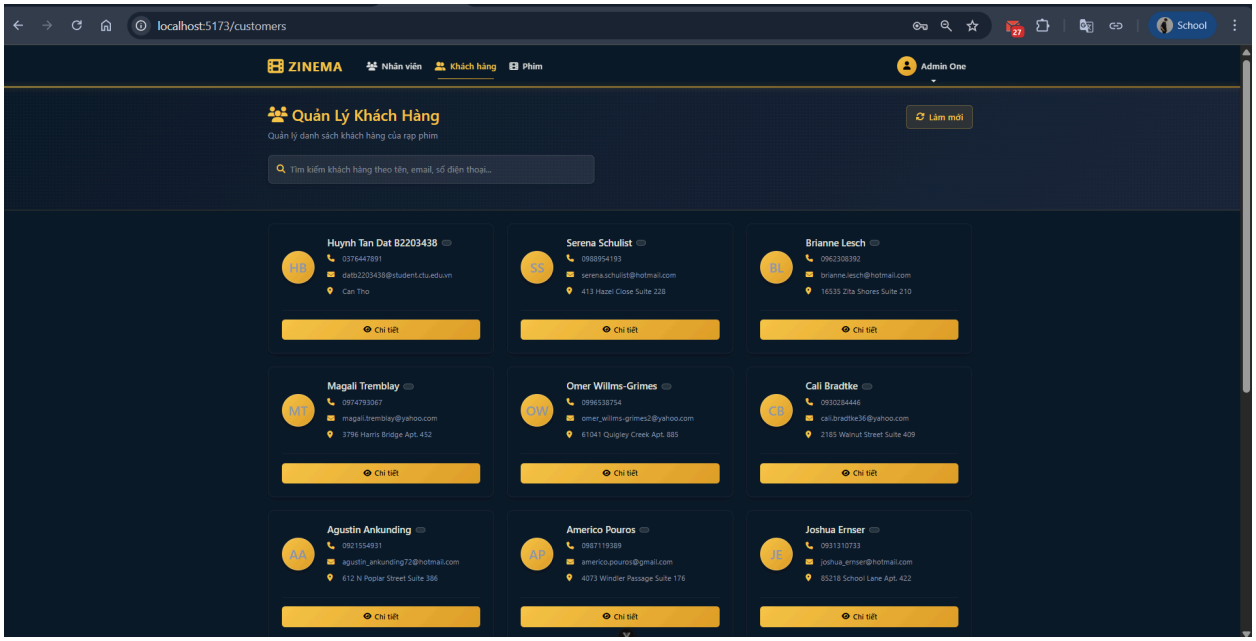


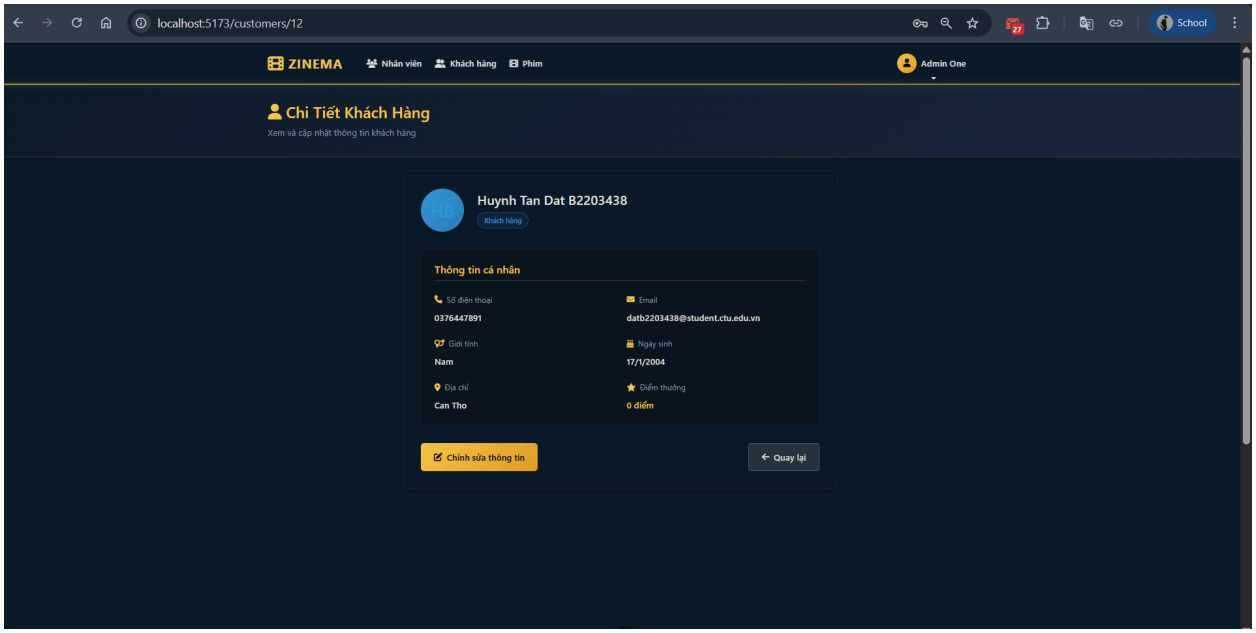
- Implementation details:
  - Relies on the same stack: the `efetch` function on the frontend, and JWT/bcryptjs on the backend for secure operations.
  - Which server-side APIs does this feature use:
    - Get All Employees (Admin only):
      - Endpoint: GET /api/auth/employees
      - Data sent: JWT token with admin role in Authorization header. Optional query parameters for pagination (page, limit).

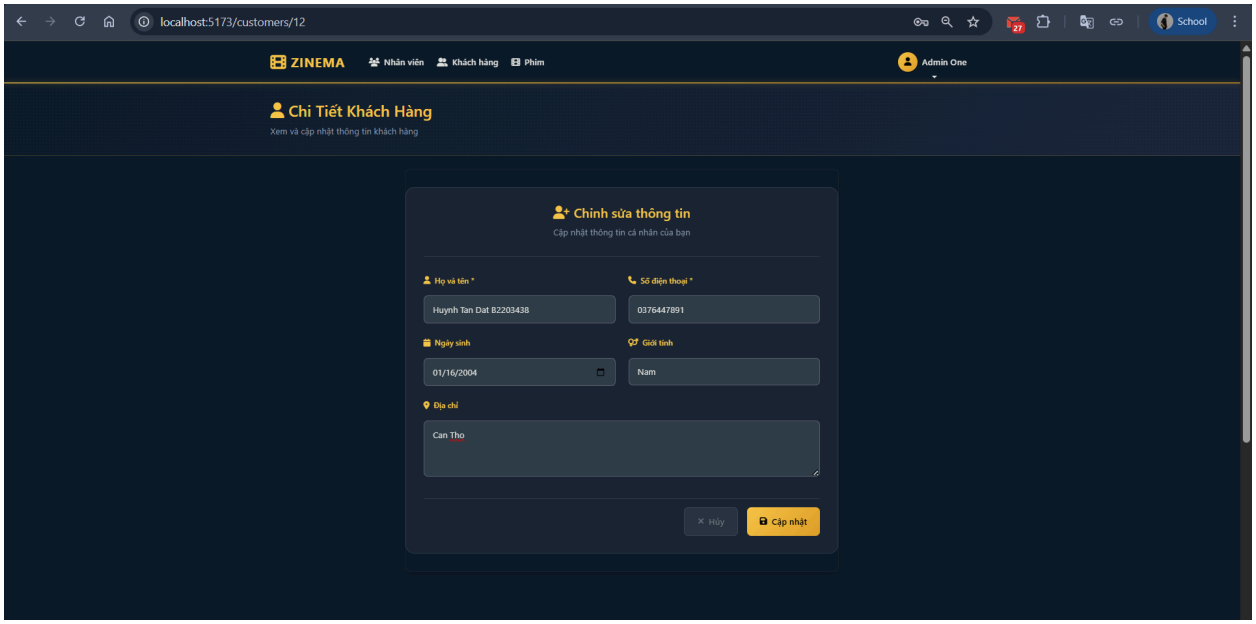
        - Response: A JSON object containing an array of employee records and pagination metadata.
    - Get Employee by ID (Admin only):
        - Endpoint: GET /api/auth/employees/:id
        - Data sent: Employee ID in URL path parameter and JWT token with admin role in Authorization header.
        - Response: A JSON object containing the profile information for a single employee.
    - Create Employee Account (Admin only):
        - Endpoint: POST /api/auth/employee/register
        - Data sent: JWT token with admin role in Authorization header, and a request body (JSON) containing phone_number, password, password_confirm, full_name, email, and optional fields like date_of_birth, address, and position.
        - Response: A JSON object containing the newly created employee's information.
    - Update Employee Account (Admin only):
        - Endpoint: PUT /api/auth/employees/:id
        - Data sent: Employee ID in URL path parameter, JWT token with admin role in Authorization header, and a request body (JSON) with fields to update.
        - Response: A JSON object containing the updated employee's information.
- Does this feature read/store data? In which table?
    - It reads from and writes to both the employees table (for profile information) and the accounts table (for login credentials). It also reads from the roles table to assign the 'staff' role.
- Which client-side states are needed to implement this feature?
    - employees: An array holding the list of all employees.
    - employee: An object for the form used to add or edit an employee.
    - isLoading, error: Local component state for user feedback during API requests.

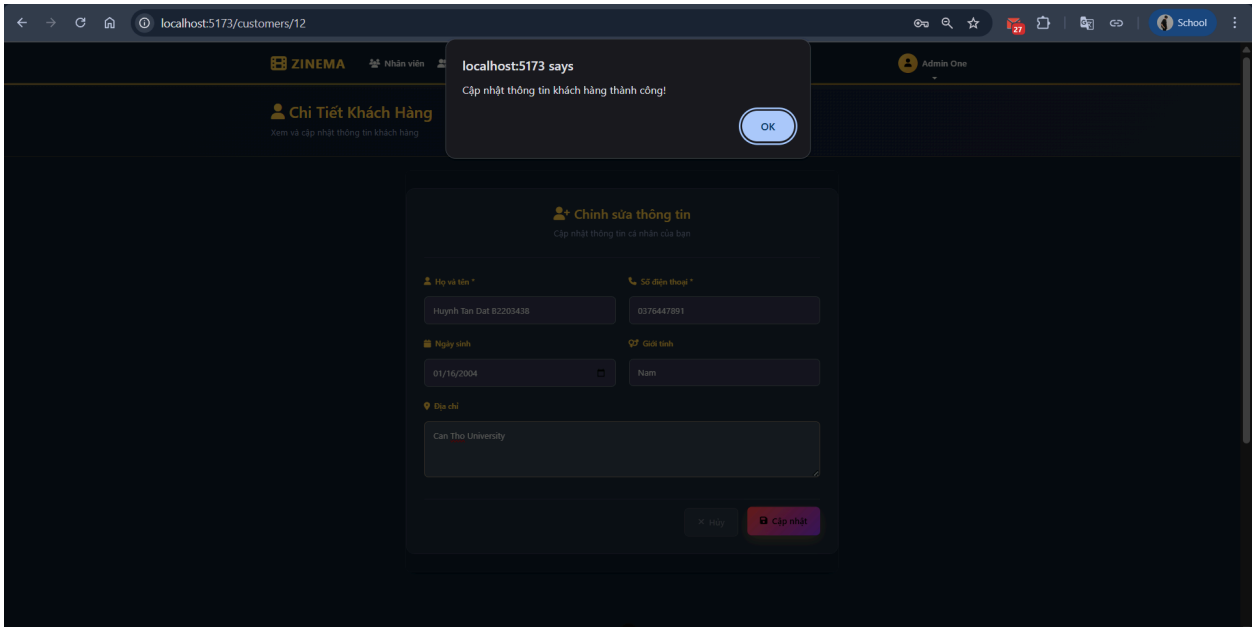
4. Customer Management

- Description: This feature allows administrators to view and manage customer accounts. Admins can see a list of all registered customers and view/update their profile details.
- Screenshots:







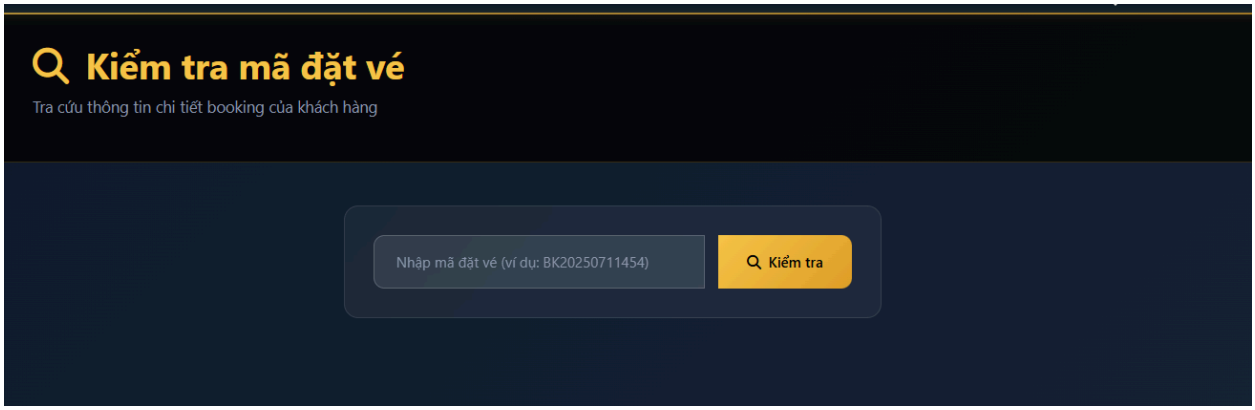


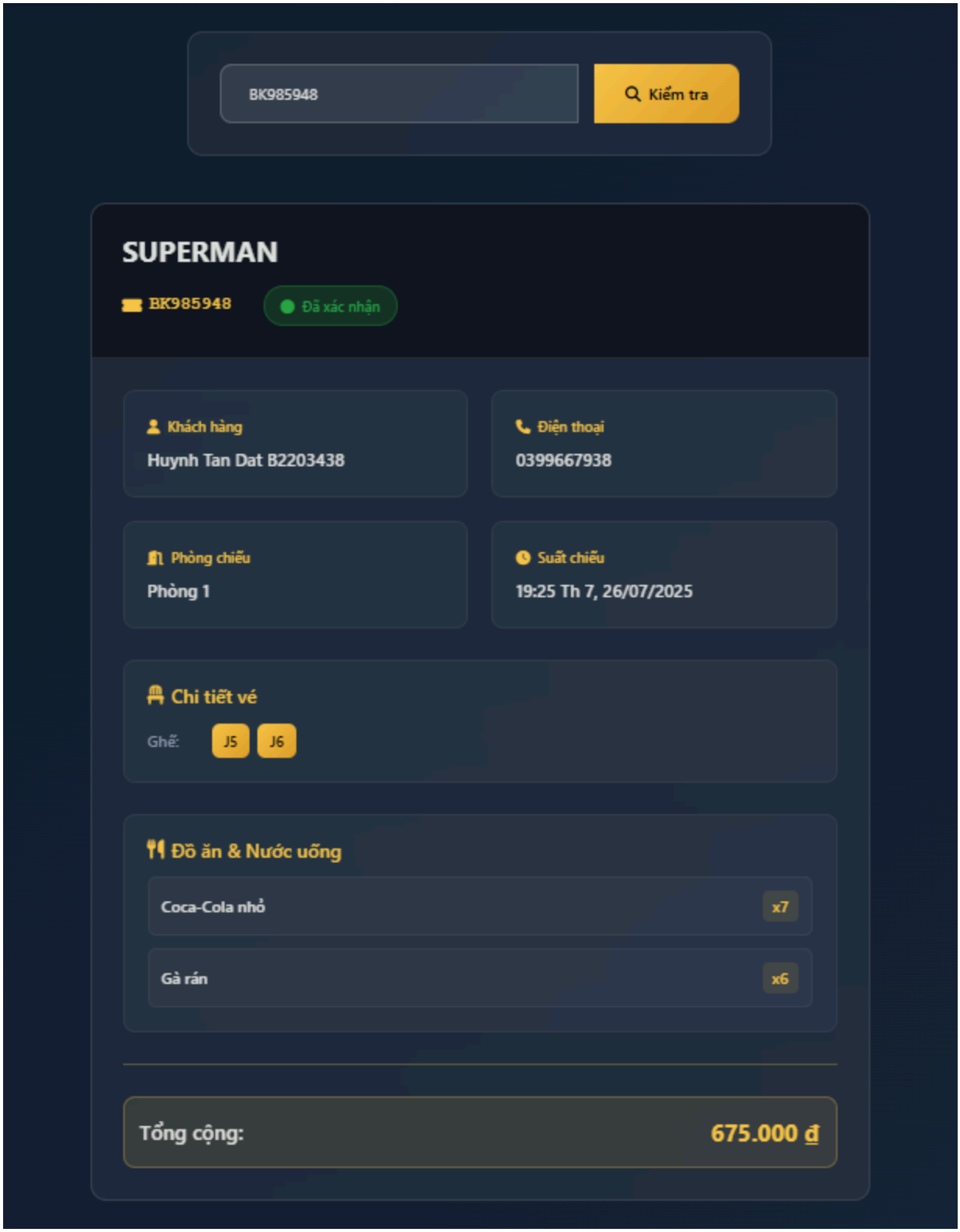
- Implementation details:
  - Standard implementation using the `efetch` function for API calls from the admin dashboard to the secure backend endpoints.
  - Which server-side APIs does this feature use:
    - Get All Customers (Admin only):
      - Endpoint: GET /api/auth/customers
      - Data sent: JWT token with admin role in Authorization header. Optional query parameters for pagination (page, limit).

        - Response: A JSON object containing an array of customer records and pagination metadata.
    - Get Customer by ID (Admin only for other users):
        - Endpoint: GET /api/auth/customers/:id
        - Data sent: Customer ID in URL path parameter and JWT token in Authorization header.
        - Response: A JSON object containing the profile information for a single customer.
    - Update Customer Profile (Admin only for other users):
        - Endpoint: PUT /api/auth/customers/:id
        - Data sent: Customer ID in URL path parameter, JWT token in Authorization header, and a request body (JSON) with fields to update.
        - Response: A JSON object containing the updated customer's information.
- Does this feature read/store data? In which table?
    - It reads from and writes to the customers table for profile information and the accounts table to retrieve associated account details.
- Which client-side states are needed to implement this feature?
    - customers: An array storing the list of all customers.
    - customer: An object to hold the data for the customer being viewed or edited.
    - pagination: An object for managing the pages of the customer list.
    - isLoading, error: State variables for handling API call status.

5. Booking Checking

- Description: This feature enables cinema staff to look up and verify booking information. Staff can search for a booking using either the booking ID or the customer's phone number to retrieve the full details of the reservation, including movie, showtime, seats, and food items.
- Screenshots:


Kiểm tra mã đặt vé
Tra cứu thông tin chi tiết booking của khách hàng
Nhập mã đặt vé (ví dụ: BK20250711454)
Kiểm tra

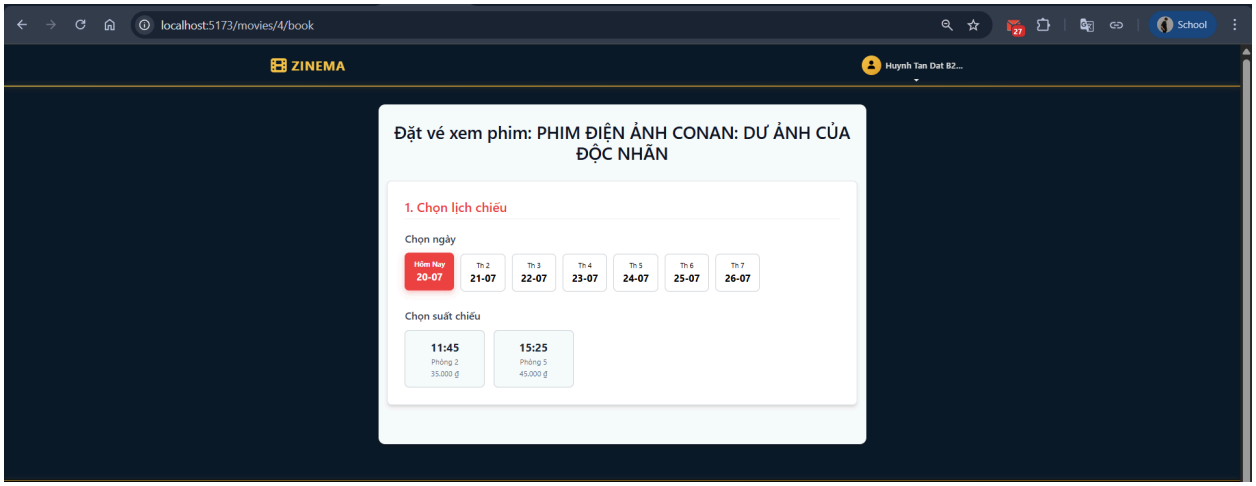


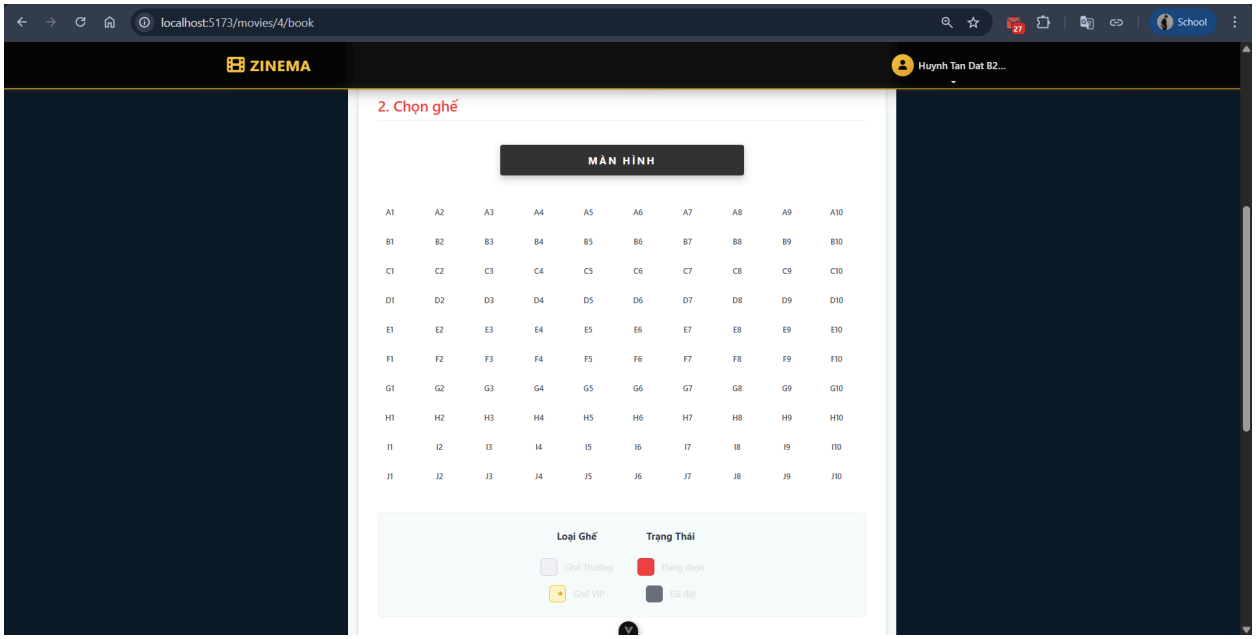
- Implementation details:

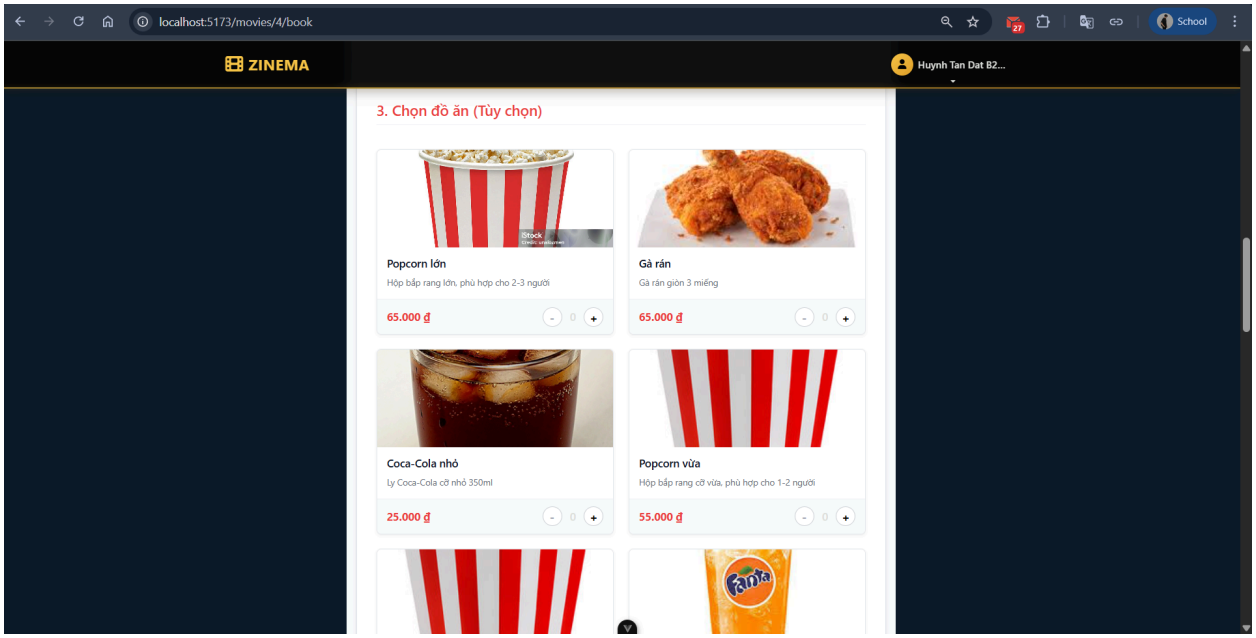
- A simple form on the frontend sends a query to the backend using the `efetch` function.
- Which server-side APIs does this feature use:
  - Get Booking By Code (For Staff)
    - Endpoint: GET /api/bookings/code/:code
    - Data sent: The booking code is sent as a URL path parameter (:code), and a JWT token for a staff or admin role is required in the Authorization header.
    - Response: A JSON object containing the detailed information of the found booking, including customer data, movie details, showtime, seats, and purchased food items.
- Does this feature read/store data? In which table?
  - This feature is read-only. It reads data from multiple tables to assemble the booking details: bookings, customers, movies, showtimes, cinema_rooms, tickets, seats, food_bookings, and foods.
- Which client-side states are needed to implement this feature?
  - searchTerm: A string to hold the input from the staff (booking ID or phone number).
  - foundBooking: An object to store the booking details returned from the API.
  - isLoading, error: Local state for feedback during the search.
  - searchType: A state to toggle between searching by ID or by phone number.

6. Showtime, Seat, and Food Selection

- Description: This feature set allows customers to plan their visit after selecting a movie. It includes selecting a showtime for a specific movie, viewing the seating map for that showtime to pick available seats, and choosing food and beverage items to add to their booking.
- Screenshots:


localhost:5173/movies/4/book
School
ZINEMA
Huynh Tan Dat B2...
Đặt vé xem phim: PHIM ĐIỆN ẢNH CONAN: DƯ ẢNH CỦA ĐỘC NHÃN
1. Chọn lịch chiếu
Chọn ngày
Hôm Nay 20-07
Th 2 21-07
Th 3 22-07
Th 4 23-07
Th 5 24-07
Th 6 25-07
Th 7 26-07
Chọn suất chiếu
11:45
Phòng 2
35.000 ₫
15:25
Phòng 5
45.000 ₫


localhost:5173/movies/4/book
School
ZINEMA
Huynh Tan Dat B2...
2. Chọn ghế
MÀN HÌNH
A1 A2 A3 A4 A5 A6 A7 A8 A9 A10
B1 B2 B3 B4 B5 B6 B7 B8 B9 B10
C1 C2 C3 C4 C5 C6 C7 C8 C9 C10
D1 D2 D3 D4 D5 D6 D7 D8 D9 D10
E1 E2 E3 E4 E5 E6 E7 E8 E9 E10
F1 F2 F3 F4 F5 F6 F7 F8 F9 F10
G1 G2 G3 G4 G5 G6 G7 G8 G9 G10
H1 H2 H3 H4 H5 H6 H7 H8 H9 H10
I1 I2 I3 I4 I5 I6 I7 I8 I9 I10
J1 J2 J3 J4 J5 J6 J7 J8 J9 J10
Loại Ghế
Trạng Thái
Ghế Thường
Đang chọn
Ghế VIP
Đã đặt



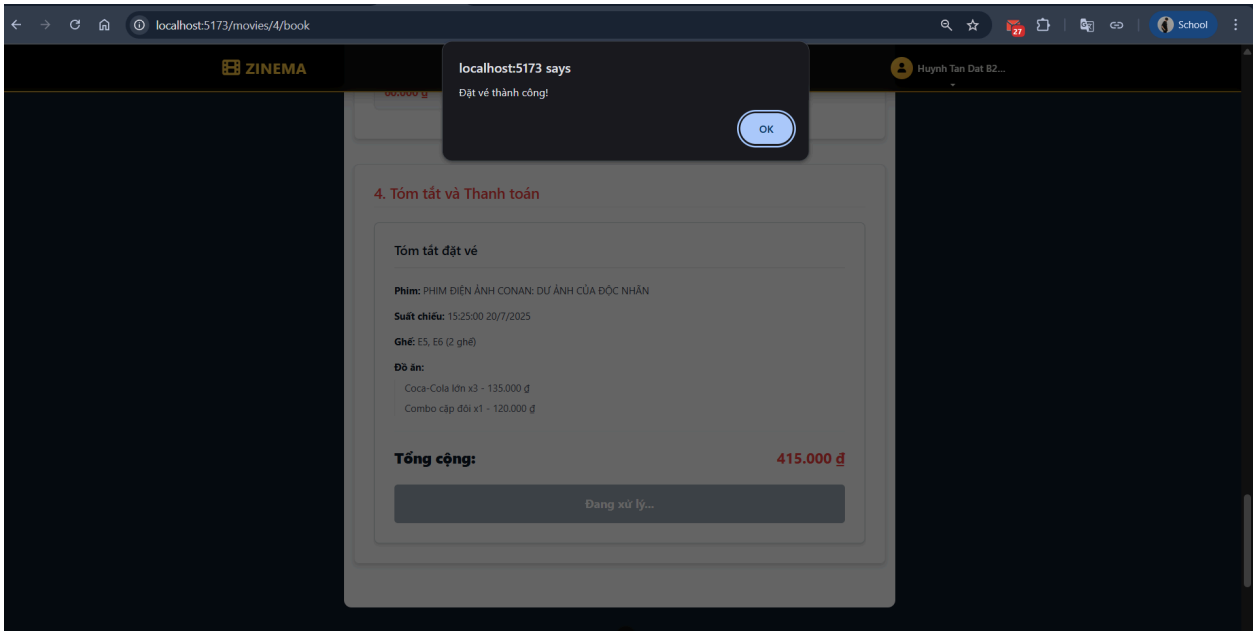
- Implementation details:
  - This feature is a multi-step process managed on the client-side, using Vue Router or component logic to guide the user. The `efetch` function fetches the necessary data for each step.
  - Which server-side APIs does this feature use:
    - Get Showtimes for a Movie:
      - Endpoint: GET /api/showtimes/movie/:movieId
      - Data sent: Movie ID in the URL path parameter.
      - Response: A JSON object containing an array of available showtimes for the specified movie.
    - Get Seat Map for a Showtime:
      - Endpoint: GET /api/showtimes/:id/seats
      - Data sent: Showtime ID in the URL path parameter.
      - Response: A JSON object containing an array of all seats for the showtime's cinema room, with an is_booked status for each seat.
    - Get All Food Items:
      - Endpoint: GET /api/foods
      - Data sent: Optional query parameters for pagination (page, limit).
      - Response: A JSON object containing an array of all available food and beverage items.
  - Does this feature read/store data? In which table?

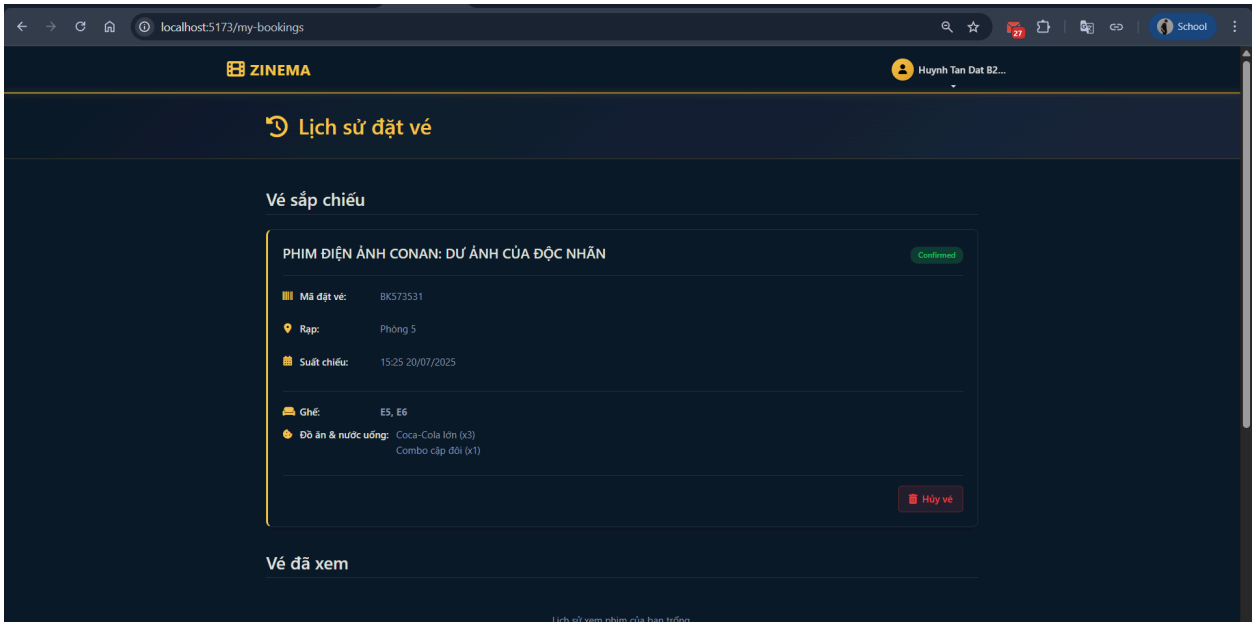
- It reads from the showtimes and movies tables to display screening times.
- It reads from the seats, cinema_rooms, and tickets tables to generate the seating map and determine seat availability.
- It reads from the foods table to display the menu of available items.

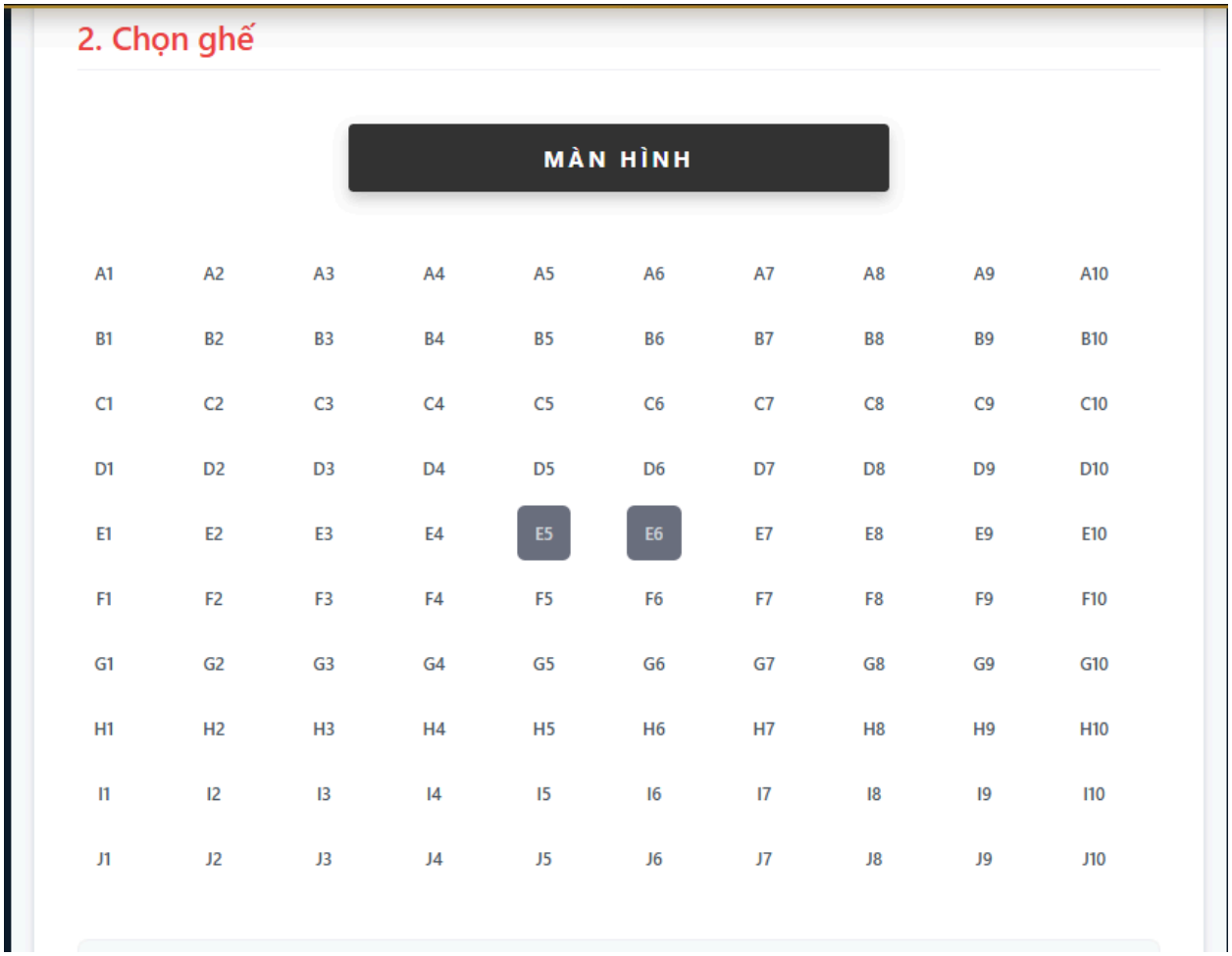
- Which client-side states are needed to implement this feature?
  - selectedMovie: The movie object the user is booking.
  - selectedShowtime: The ID of the chosen showtime.
  - selectedSeats: An array of seat objects the user has picked.
  - selectedFoods: An array of food objects with their selected quantities.
  - These states are passed between steps to build the final order.

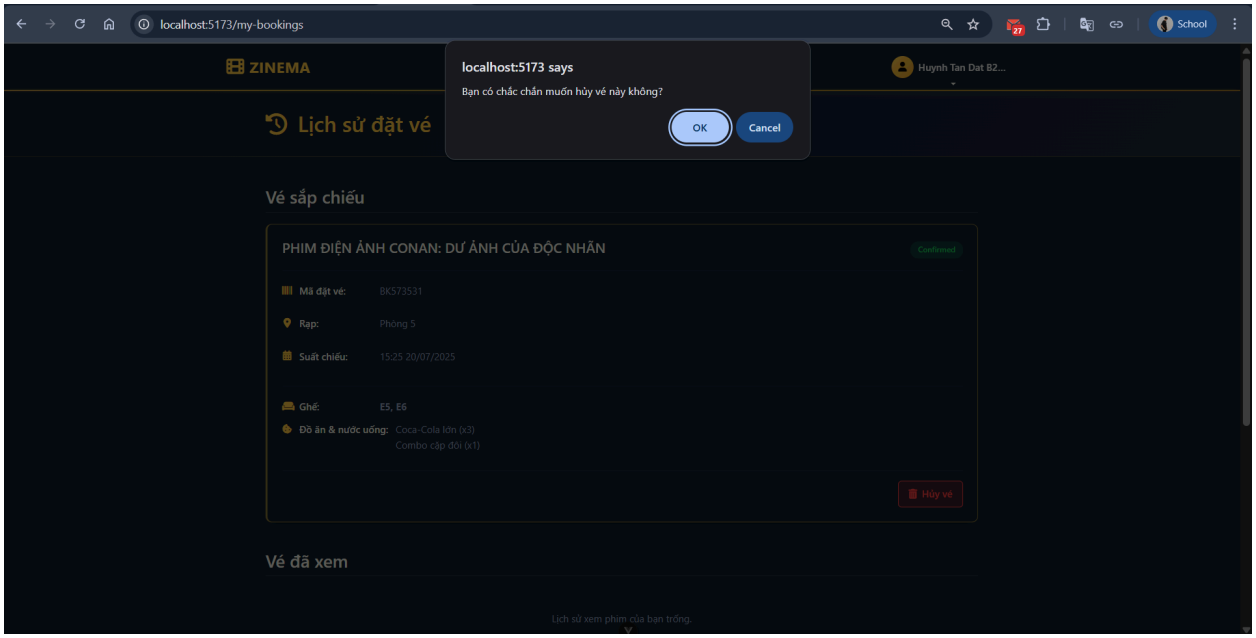
7. Booking Management

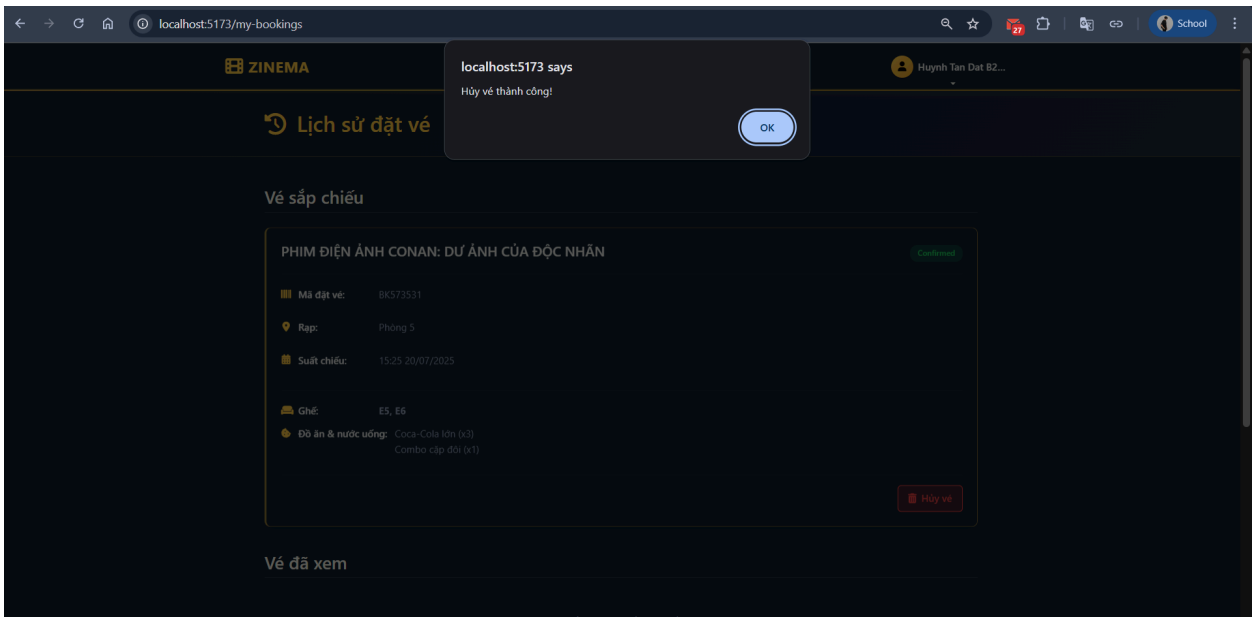
- Description: This feature enables authenticated customers to create and manage their bookings. After selecting a movie, showtime, seats, and food, the customer can finalize and pay for their booking. They can also view their booking history and cancel an upcoming booking.
- Screenshots:

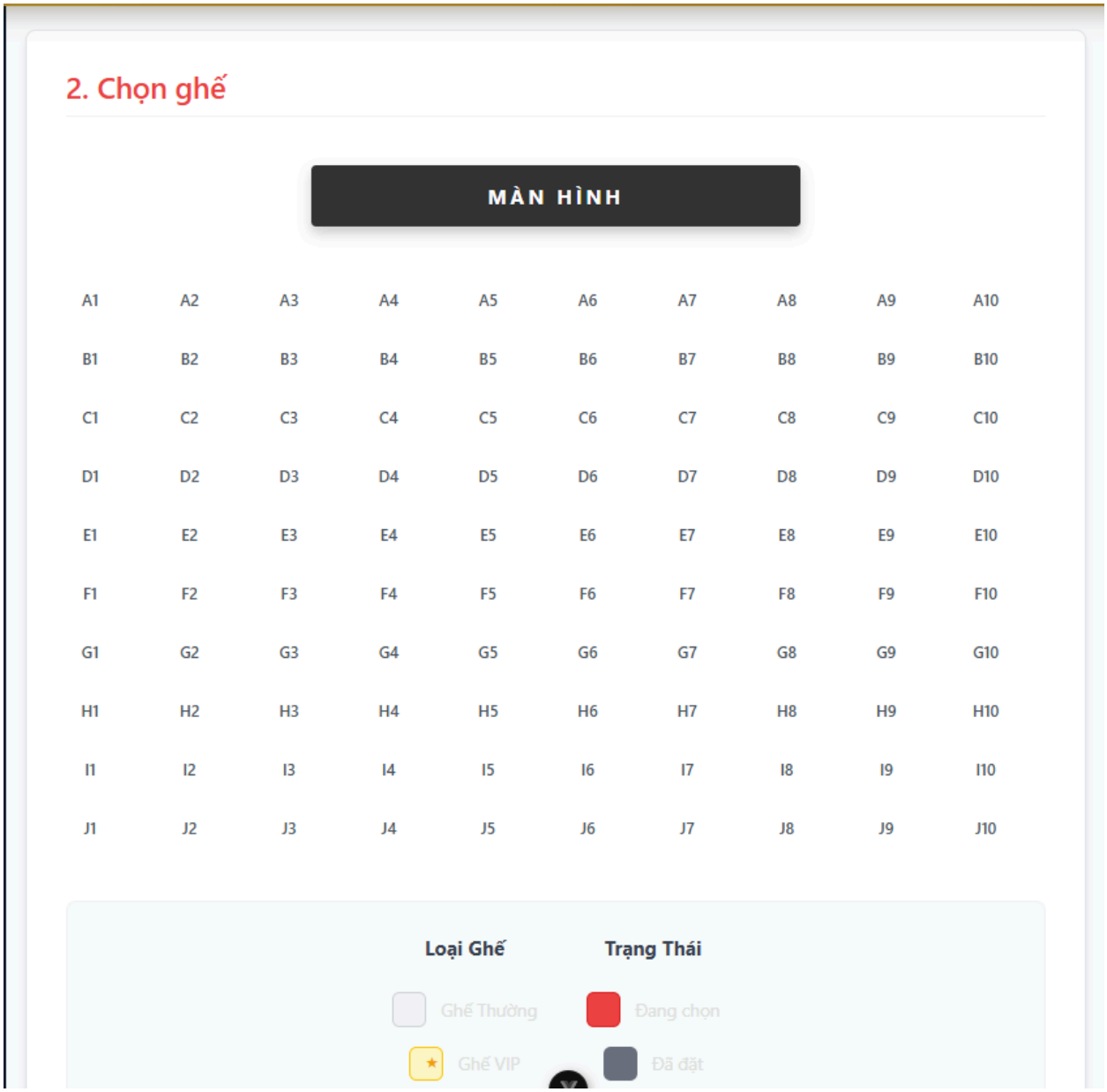



localhost:5173/my-bookings
School
ZINEMA
Huynh Tan Dat B2...
Lịch sử đặt vé
Vé sắp chiếu
PHIM ĐIỆN ẢNH CONAN: DƯ ẢNH CỦA ĐỘC NHÃN
Confirmed
Mã đặt vé: BK573531
Rạp: Phòng 5
Suất chiếu: 15:25 20/07/2025
Ghế: E5, E6
Đồ ăn & nước uống: Coca-Cola lớn (x3)
Combo cặp đôi (x1)
Hủy vé
Vé đã xem


2. Chọn ghế
MÀN HÌNH
A1 A2 A3 A4 A5 A6 A7 A8 A9 A10
B1 B2 B3 B4 B5 B6 B7 B8 B9 B10
C1 C2 C3 C4 C5 C6 C7 C8 C9 C10
D1 D2 D3 D4 D5 D6 D7 D8 D9 D10
E1 E2 E3 E4 E5 E6 E7 E8 E9 E10
F1 F2 F3 F4 F5 F6 F7 F8 F9 F10
G1 G2 G3 G4 G5 G6 G7 G8 G9 G10
H1 H2 H3 H4 H5 H6 H7 H8 H9 H10
I1 I2 I3 I4 I5 I6 I7 I8 I9 I10
J1 J2 J3 J4 J5 J6 J7 J8 J9 J10


localhost:5173/my-bookings
ZINEMA
Huynh Tan Dat B2...
localhost:5173 says
Bạn có chắc chắn muốn hủy vé này không?
OK
Cancel
Lịch sử đặt vé
Vé sắp chiếu
PHIM ĐIỆN ẢNH CONAN: DƯ ẢNH CỦA ĐỘC NHÃN
Confirmed
Mã đặt vé: BK573531
Rạp: Phòng 5
Suất chiếu: 15:25 20/07/2025
Ghế: E5, E6
Đồ ăn & nước uống: Coca-Cola lớn (x3)
Combo cặp đôi (x1)
Hủy vé
Vé đã xem
Lịch sử xem phim của bạn trống.


localhost:5173/my-bookings
ZINEMA
Huynh Tan Dat B2...
localhost:5173 says
Hủy vé thành công!
OK
Lịch sử đặt vé
Vé sắp chiếu
PHIM ĐIỆN ẢNH CONAN: DƯ ẢNH CỦA ĐỘC NHÃN
Confirmed
Mã đặt vé: BK573531
Rạp: Phòng 5
Suất chiếu: 15:25 20/07/2025
Ghế: E5, E6
Đồ ăn & nước uống: Coca-Cola lớn (x3)
Combo cặp đôi (x1)
Hủy vé
Vé đã xem



- Implementation details:
  - Uses the state built in the previous feature to send a final creation request via the `efetch` function. The booking history is a simple fetch-and-display list.
  - Which server-side APIs does this feature use:
    - Create Booking:
      - Endpoint: POST /api/bookings

        - Data sent: JWT token in Authorization header. Request body (JSON) includes showtime_id, an array of seat_ids, and an array of foods (with food_id and quantity).
        - Response: A JSON object containing the details of the newly created booking, including the booking ID.
    - Get My Bookings:
        - Endpoint: GET /api/bookings/my-bookings
        - Data sent: JWT token in Authorization header.
        - Response: A JSON object containing an array of all bookings made by the logged-in customer.
    - Cancel Booking:
        - Endpoint: DELETE /api/bookings/:id
        - Data sent: Booking ID in the URL path parameter and JWT token in Authorization header.
        - Response: A JSON success message confirming the cancellation.
- Does this feature read/store data? In which table?
    - It writes new records to the bookings, tickets, and food_bookings tables when a booking is created.
    - It reads from these same tables, along with movies and showtimes, to display the user's booking history.
    - The delete operation updates the status in the bookings table to 'cancelled' and deletes the associated tickets and food_bookings records.
- Which client-side states are needed to implement this feature?
    - The states from the selection feature (selectedShowtime, selectedSeats, etc.) are consumed here.
    - myBookings: An array to store the user's booking history.
    - isLoading, error: Local state for user feedback during booking confirmation or cancellation.