ADMIN PANEL CINEBOOK - SCRIPT THUYẾT TRÌNH CHI TIẾT

PHẦN 1: TỔNG QUAN KIẾN TRÚC VÀ LUỒNG HOẠT ĐỘNG

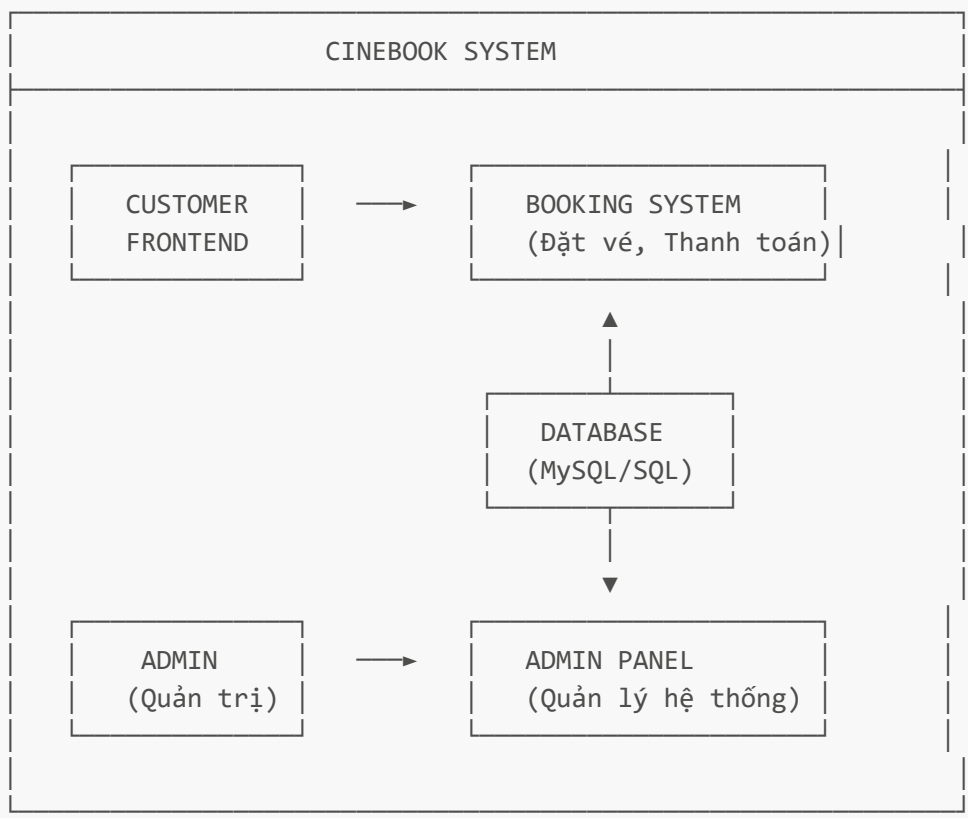
MỤC LỤC SERIES TÀI LIỆU

1. GIỚI THIỆU HỆ THỐNG ADMIN

1.1. Admin Panel là gì?

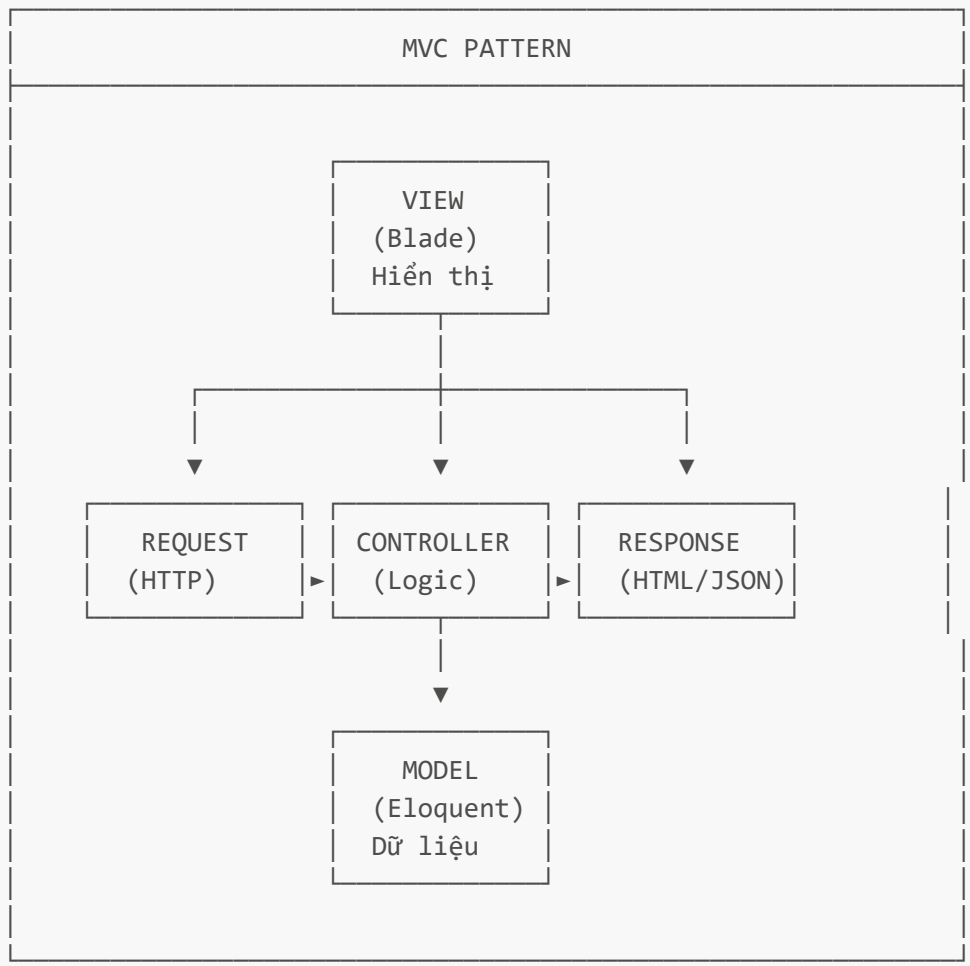
Admin Panel là giao diện quản trị dành cho người quản lý hệ thống đặt vé xem phim CineBook. Đây là nơi tập trung mọi thao tác CRUD (Create, Read, Update, Delete) cho các đối tượng chính của hệ thống.

1.2. Vai trò trong hệ thống



2. KIẾN TRÚC MVC TRONG ADMIN PANEL

2.1. Mô hình MVC (Model-View-Controller)



2.2. Cấu trúc thư mục Admin

```
app/
├── Http/
│   ├── Controllers/
│   │   └── Admin/                          # Admin Controllers
│   │       ├── AdminDashboardController.php   # Dashboard
│   │       ├── AdminMovieController.php       # Quản lý phim
│   │       ├── AdminUserController.php        # Quản lý user
│   │       ├── AdminRoomController.php        # Quản lý phòng chiếu
│   │       ├── AdminShowtimeController.php    # Quản lý suất chiếu
│   │       ├── AdminBookingController.php     # Quản lý đơn đặt vé
│   │       ├── AdminReviewController.php      # Quản lý đánh giá
│   │       └── QRCheckInController.php        # Check-in QR
│   │
│   └── Middleware/
│       └── CheckRole.php                   # Middleware phân quyền
│
├── Models/                                 # Eloquent Models
```

```
│   ├── User.php
│   ├── Movie.php
│   ├── Room.php
│   ├── Seat.php
│   ├── Showtime.php
│   ├── Booking.php
│   └── BookingSeat.php
│
resources/
└── views/
    ├── admin/                              # Admin Views
    │   ├── layouts/
    │   │   └── admin.blade.php             # Layout chính
    │   ├── dashboard.blade.php             # Trang Dashboard
    │   ├── movies/                         # Views quản lý phim
    │   ├── users/                          # Views quản lý user
    │   ├── rooms/                          # Views quản lý phòng
    │   ├── showtimes/                      # Views suất chiếu
    │   ├── bookings/                       # Views đơn đặt vé
    │   ├── reviews/                        # Views đánh giá
    │   └── qr_checkin/                     # Views check-in QR
    │
    └── layouts/
        └── admin.blade.php                 # Master layout

routes/
└── web.php                                 # Route definitions
```

---

3. HỆ THỐNG PHÂN QUYỀN (AUTHORIZATION)

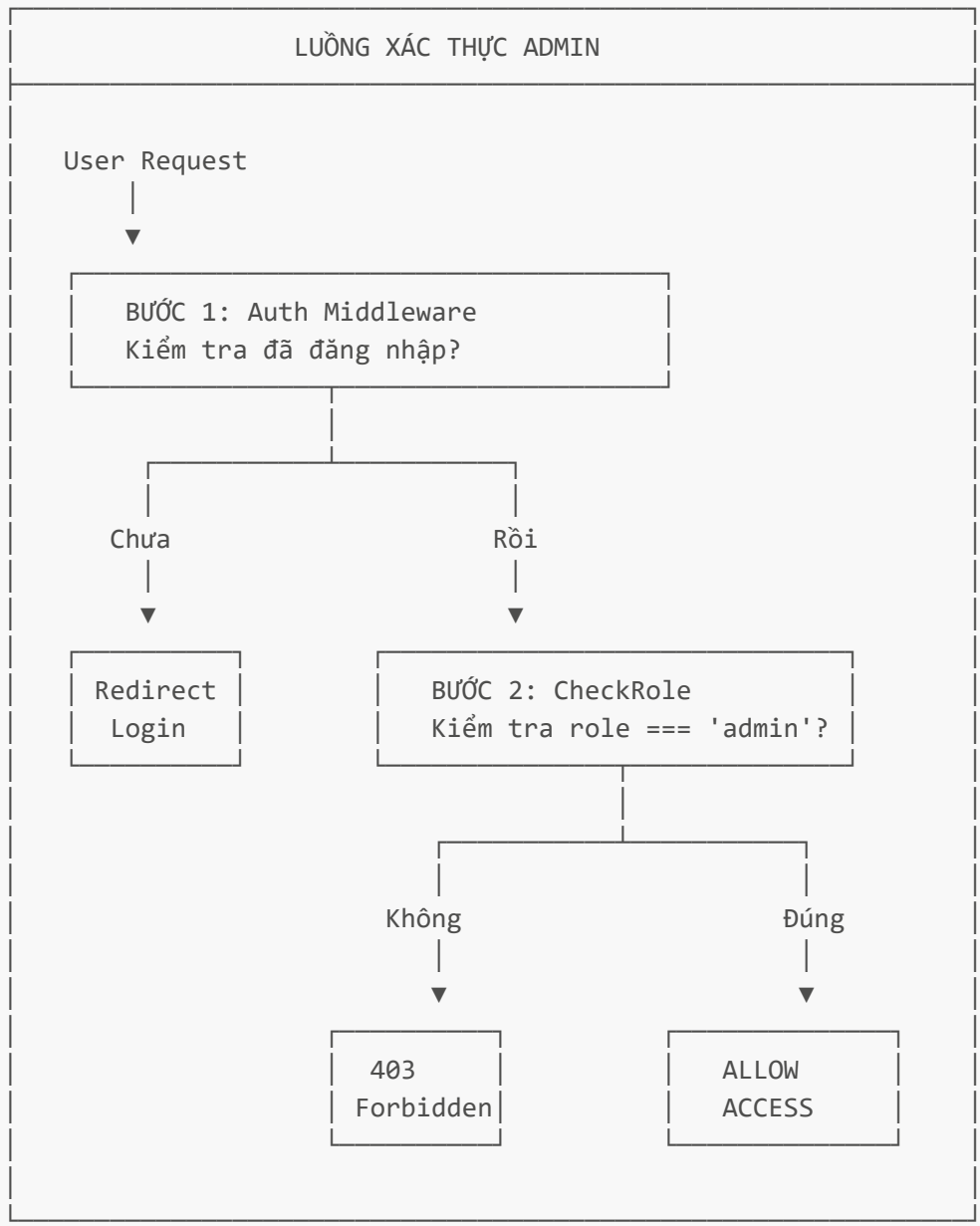
3.1. Middleware CheckRole

File: `app/Http/Middleware/CheckRole.php`

```php
class CheckRole
{
    public function handle(Request $request, Closure $next, string $role)
    {
        // Bước 1: Kiểm tra đã đăng nhập chưa
        if (!Auth::check()) {
            return redirect()->route('login')
                ->with('error', 'You need to log in to continue');
        }

        // Bước 2: Kiểm tra role có đúng không
        if (Auth::user()->role !== $role) {
            abort(403, 'You do not have permission to access this page');
        }

        // Bước 3: Cho phép tiếp tục
```

```
            return $next($request);
        }
    }
```

3.2. Luồng xác thực (Authentication Flow)



3.3. Route Protection

File: `routes/web.php`

```
// Tất cả route admin được bảo vệ bởi 2 middleware
Route::prefix('admin')
    ->middleware(['auth', 'role:admin'])  // ← Bảo vệ ở đây
    ->group(function () {
```

```
        Route::get('/', [AdminDashboardController::class, 'index'])
            ->name('admin.dashboard');

        Route::resource('movies', AdminMovieController::class, ['as' => 'admin']);
        Route::resource('users', AdminUserController::class, ['as' => 'admin']);
        // ... các route khác
    });
```

3.4. User Model - Role Checking

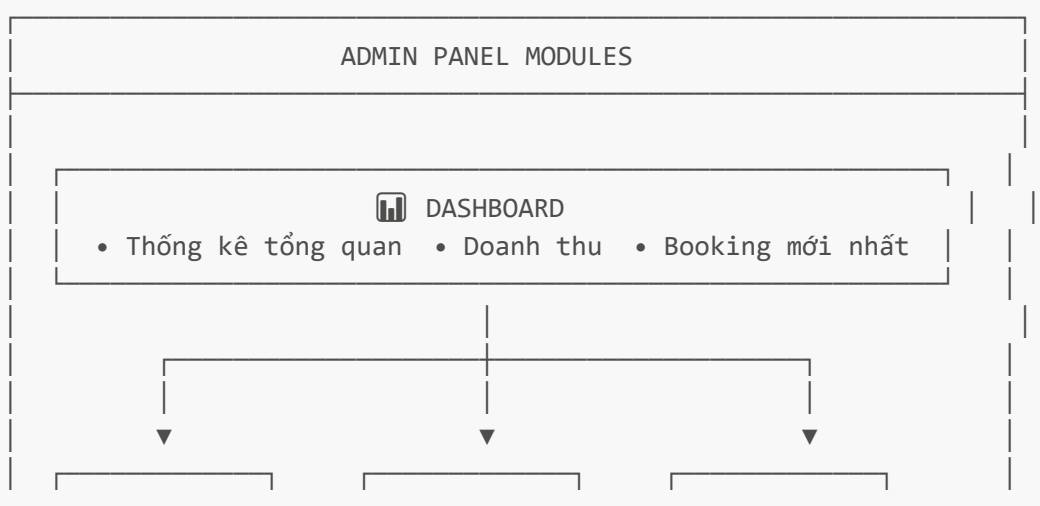
File: `app/Models/User.php`

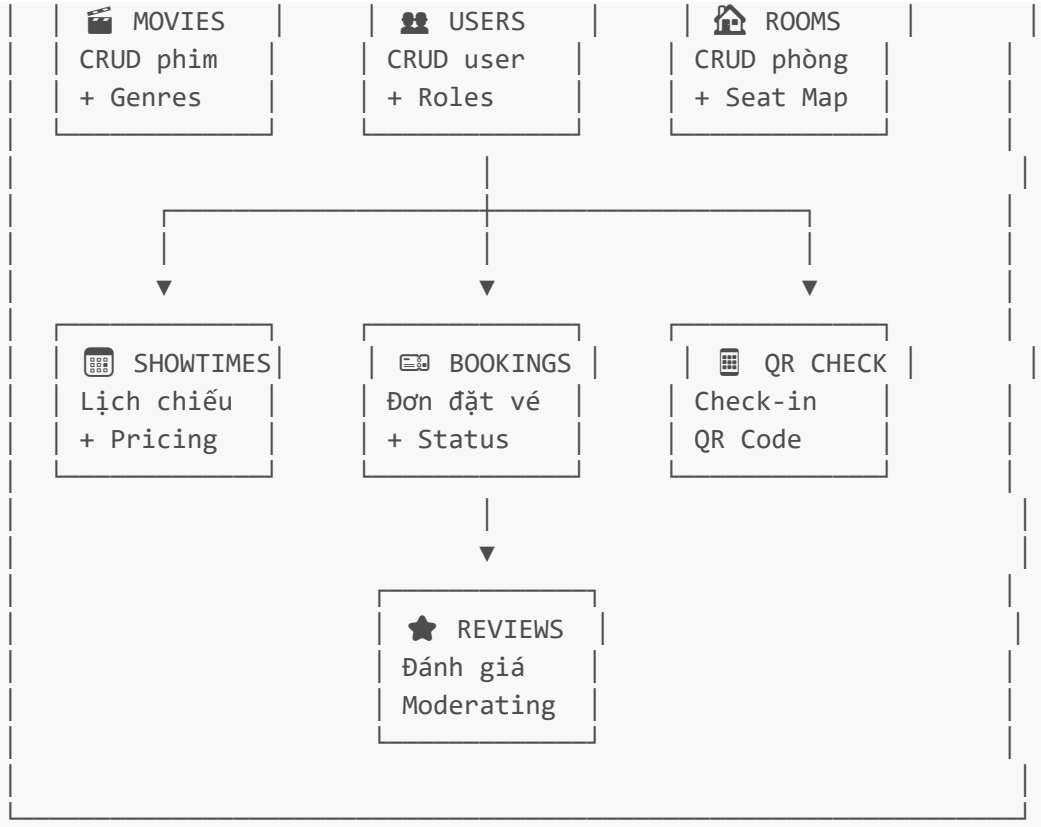
```
class User extends Authenticatable
{
    protected $fillable = [
        'name', 'email', 'password', 'city', 'phone', 'role',
    ];

    // Kiểm tra có phải admin không
    public function isAdmin()
    {
        return $this->role === 'admin';
    }

    // Kiểm tra có phải user thường không
    public function isUser()
    {
        return $this->role === 'user';
    }
}
```

---

4. CÁC MODULE CHỨC NĂNG

4.1. Sơ đồ tổng quan các Module





4.2. Dashboard Module

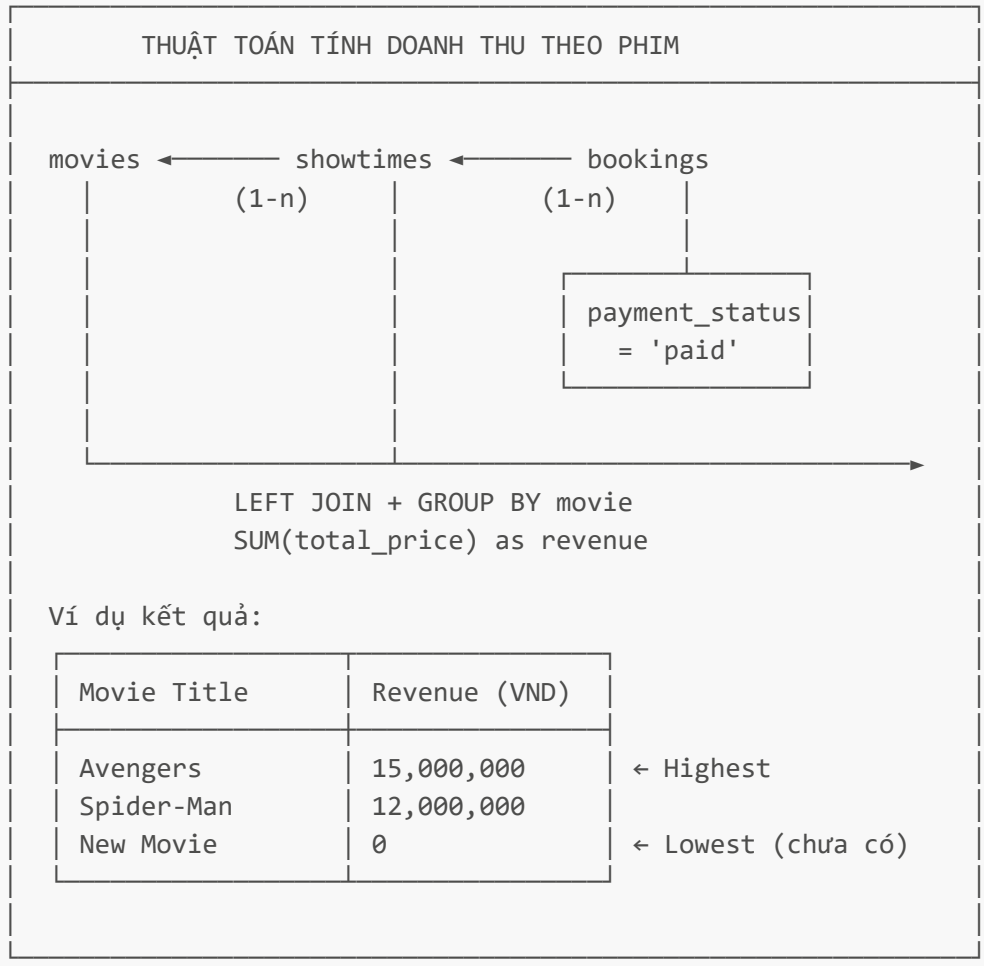
Chức năng: Hiển thị tổng quan hệ thống, thống kê quan trọng

Controller: `AdminDashboardController.php`

```php
public function index()
{
    // === THỐNG KÊ CƠ BẢN ===
    $totalUsers = User::where('role', 'user')->count();
    $totalMovies = Movie::count();

    // === THỐNG KÊ HÔM NAY ===
    $today = Carbon::today();

    // Đếm vé bán hôm nay (dùng JOIN để đếm từ booking_seats)
    $ticketsSoldToday = DB::table('booking_seats')
        ->join('bookings', 'booking_seats.booking_id', '=', 'bookings.id')
        ->whereDate('bookings.created_at', $today)
        ->where('bookings.payment_status', 'paid')
        ->count();

    // Doanh thu hôm nay
    $revenueToday = Booking::whereDate('created_at', $today)
        ->where('payment_status', 'paid')
        ->sum('total_price');

    // === PHIM DOANH THU CAO/THẤP NHẤT ===
```

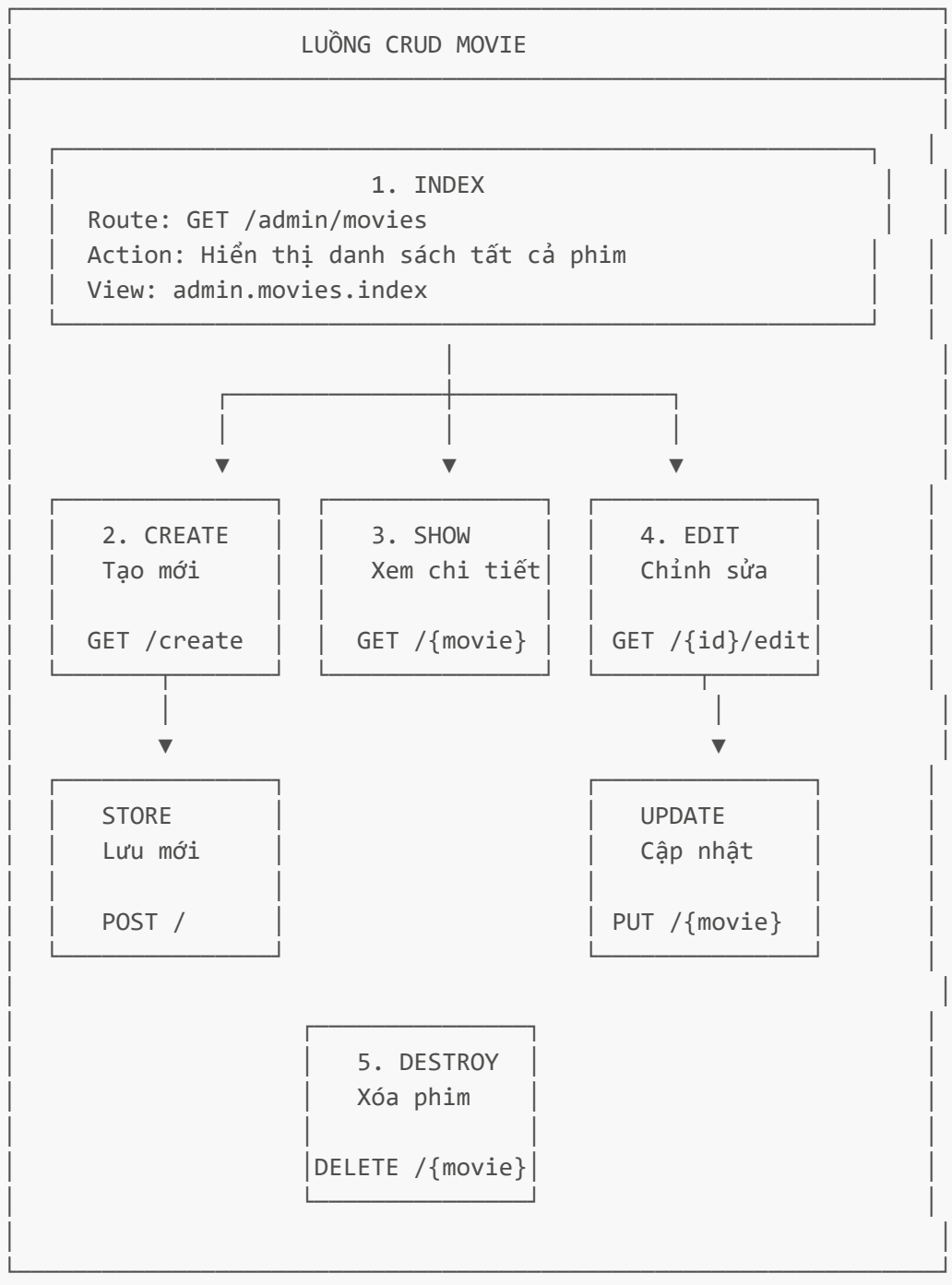
```
    $movieRevenue = Movie::leftJoin('showtimes', 'movies.id', '=',
'showtimes.movie_id')
        ->leftJoin('bookings', function ($join) {
            $join->on('showtimes.id', '=', 'bookings.showtime_id')
                ->where('bookings.payment_status', 'paid');
        })
        ->select('movies.id', 'movies.title',
                 DB::raw('COALESCE(SUM(bookings.total_price), 0) as revenue'))
        ->groupBy('movies.id', 'movies.title')
        ->orderBy('revenue', 'desc')
        ->get();

    // === BOOKING GẦN NHẤT ===
    $recentBookings = Booking::with(['user', 'showtime.movie', 'bookingSeats'])
        ->latest()
        ->take(10)
        ->get();

    return view('admin.dashboard', compact(...));
}
```

Giải thích thuật toán tính doanh thu phim:



5. LUỒNG HOẠT ĐỘNG CHÍNH

5.1. Luồng CRUD Movie (Ví dụ điển hình)



5.2. Chi tiết Store Movie

Controller: `AdminMovieController@store`

```php
public function store(Request $request)
{
    // === BƯỚC 1: VALIDATION ===
    $validated = $request->validate([
```

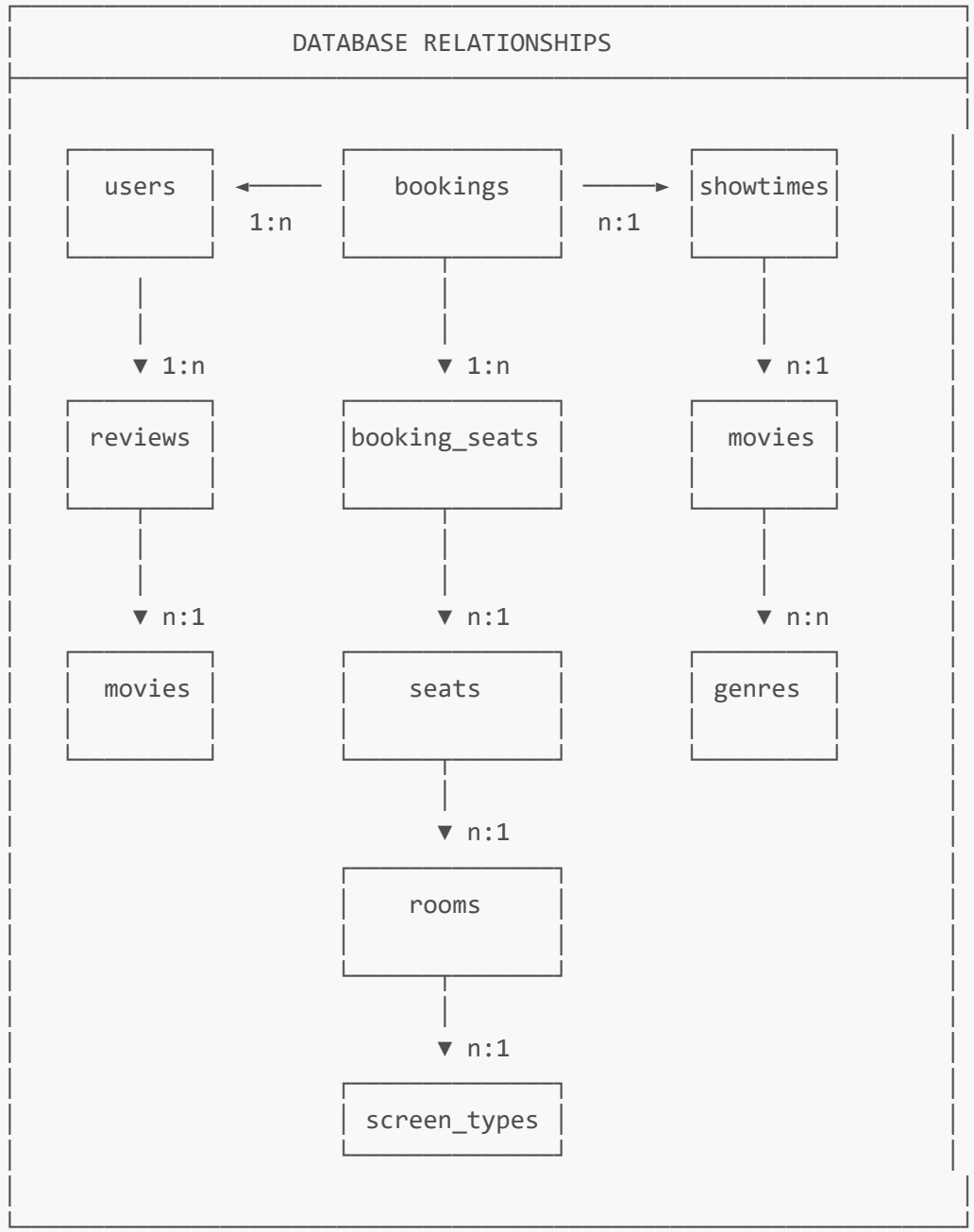
```
            'title' => 'required|string|max:255',
            'director' => 'nullable|string|max:255',
            'cast' => 'nullable|string',
            'genres' => 'nullable|array',
            'genres.*' => 'exists:genres,id',      // Kiểm tra genre tồn tại
            'duration' => 'required|integer|min:1',
            'release_date' => 'required|date',
            'rating' => 'nullable|numeric|min:0|max:10',
            'poster_url' => 'nullable|url',
            'trailer_url' => 'nullable|url',
            'status' => 'required|in:now_showing,coming_soon,ended',
        ]);

        // === BƯỚC 2: TÁCH GENRES ===
        $genres = $validated['genres'] ?? [];
        unset($validated['genres']);   // Không cho vào mass assignment

        // === BƯỚC 3: TẠO MOVIE ===
        $movie = Movie::create($validated);

        // === BƯỚC 4: SYNC GENRES (Many-to-Many) ===
        if (!empty($genres)) {
            $movie->genres()->sync($genres);
        }

        // === BƯỚC 5: REDIRECT VỚI FLASH MESSAGE ===
        return redirect()->route('admin.movies.index')
            ->with('success', 'Movie created successfully!');
    }
```

Giải thích Validation Rules:

Rule	Ý nghĩa
`required`	Bắt buộc nhập
`string`	Phải là chuỗi
`max:255`	Tối đa 255 ký tự
`nullable`	Cho phép null/empty
`array`	Phải là mảng
`exists:genres,id`	ID phải tồn tại trong bảng genres
`integer`	Phải là số nguyên
`min:1`	Giá trị tối thiểu là 1
`numeric`	Phải là số (có thể thập phân)
`url`	Phải là URL hợp lệ
`in:a,b,c`	Chỉ được chọn a, b, hoặc c

6. QUAN HỆ DATABASE (ELOQUENT RELATIONSHIPS)

6.1. Sơ đồ quan hệ các bảng



6.2. Các loại Relationship trong code

1. belongsTo (n:1) - Nhiều thuộc về một

```
// Trong Booking.php
public function user()
{
    return $this->belongsTo(User::class);
    // booking.user_id → users.id
```

```
}

public function showtime()
{
    return $this->belongsTo(Showtime::class);
    // booking.showtime_id → showtimes.id
}
```

2. hasMany (1:n) - Một có nhiều

```
// Trong User.php
public function bookings()
{
    return $this->hasMany(Booking::class);
    // users.id → booking.user_id
}

// Trong Room.php
public function seats()
{
    return $this->hasMany(Seat::class);
    // rooms.id → seats.room_id
}
```

3. belongsToMany (n:n) - Nhiều với nhiều

```
// Trong Movie.php
public function genres()
{
    return $this->belongsToMany(Genre::class, 'movie_genre');
    // movies ↔ movie_genre ↔ genres
}
```

6.3. Eager Loading (Giải quyết N+1 Problem)

Vấn đề N+1:

```
// ❌ SẼ TẠO N+1 QUERY
$bookings = Booking::all();
foreach ($bookings as $booking) {
    echo $booking->user->name;      // Query thêm cho mỗi booking
    echo $booking->showtime->movie->title; // Thêm nhiều query nữa
}
```

Giải pháp Eager Loading:

```
// ☑ CHỈ TẠO 4 QUERY DÙ CÓ 1000 BOOKING
$bookings = Booking::with([
    'user',
    'showtime.movie',       // Nested eager loading
    'showtime.room',
    'bookingSeats.seat'
])->get();
```

---

7. SESSION VÀ FLASH MESSAGES

7.1. Cách sử dụng Flash Message

```
// Trong Controller
return redirect()->route('admin.movies.index')
    ->with('success', 'Movie created successfully!');

// Hoặc với error
return back()->with('error', 'Cannot delete user with existing bookings.');
```

7.2. Hiển thị trong View

```
{{-- Trong layouts/admin.blade.php --}}

@if(session('success'))
<div class="alert alert-success alert-dismissible fade show" role="alert">
    <i class="fas fa-check-circle me-2"></i>
    {{ session('success') }}
    <button type="button" class="btn-close" data-bs-dismiss="alert"></button>
</div>
@endif

@if(session('error'))
<div class="alert alert-danger alert-dismissible fade show" role="alert">
    <i class="fas fa-exclamation-circle me-2"></i>
    {{ session('error') }}
    <button type="button" class="btn-close" data-bs-dismiss="alert"></button>
</div>
@endif
```

---

8. KẾT LUẬN PHẦN 1

8.1. Những điểm quan trọng cần nhớ

1. **MVC Pattern**: Controller xử lý logic, Model thao tác DB, View hiển thị

2. **Middleware**: Bảo vệ route, kiểm tra authentication và authorization
3. **CRUD Operations**: Index, Create/Store, Show, Edit/Update, Destroy
4. **Eloquent Relationships**: belongsTo, hasMany, belongsToMany
5. **Eager Loading**: Giải quyết N+1 query problem
6. **Flash Messages**: Thông báo kết quả thao tác

8.2. Tiếp theo trong Part 2

- Chi tiết về Room và Seat Management
- Thuật toán tạo sơ đồ ghế ngồi
- Hệ thống QR Check-in
- Database Transactions

Xem tiếp: Part 2 - Chi tiết kỹ thuật và thuật toán