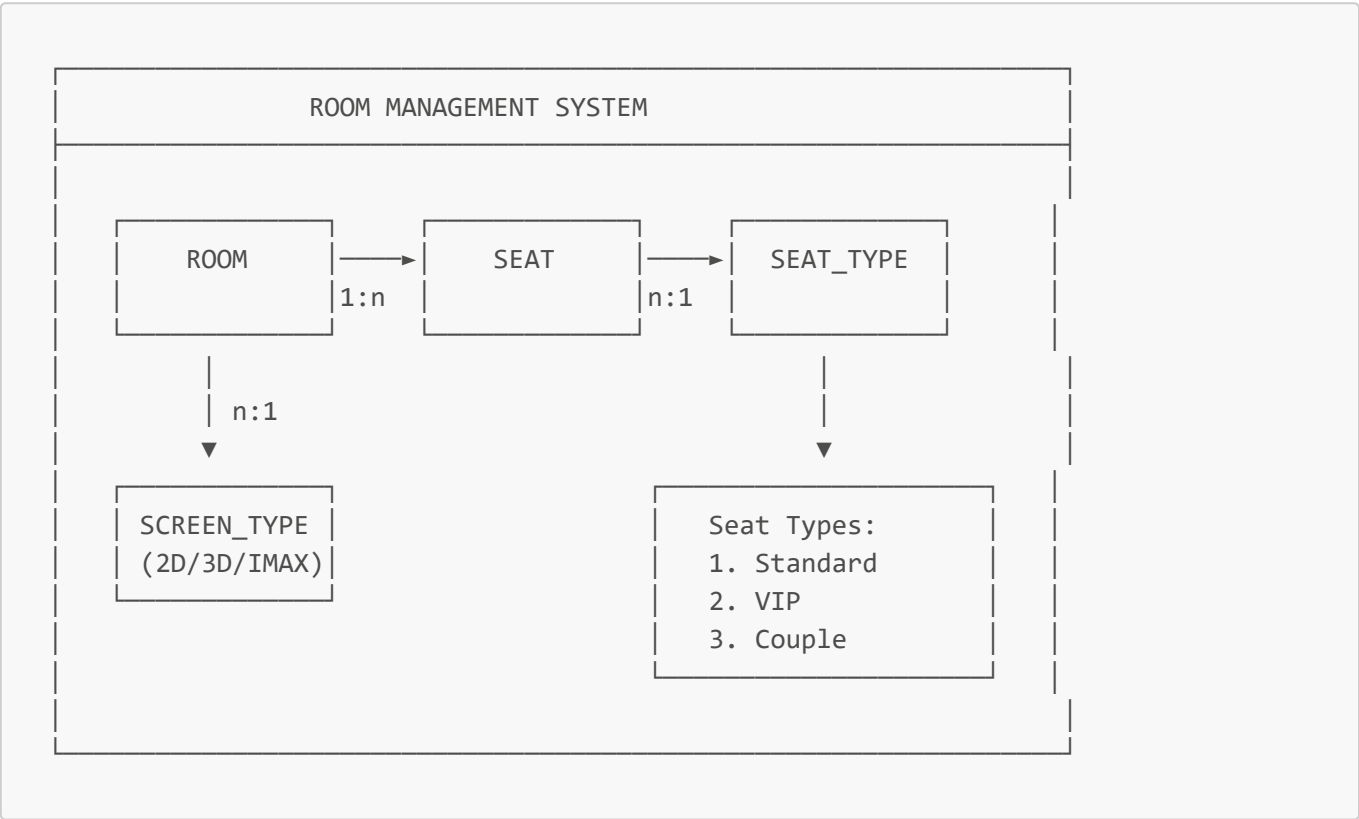


# ADMIN PANEL CINEBOOK - SCRIPT THUYẾT TRÌNH CHI TIẾT

## PHẦN 2: CHI TIẾT KỸ THUẬT VÀ THUẬT TOÁN

### 1. ROOM & SEAT MANAGEMENT - QUẢN LÝ PHÒNG CHIẾU

#### 1.1. Tổng quan hệ thống quản lý phòng



#### 1.2. Thuật toán tạo phòng và ghế

**Controller:** AdminRoomController@store

```
public function store(Request $request)
{
    // === BƯỚC 1: VALIDATION ===
    $validated = $request->validate([
        'name' => 'required|string|max:255',
        'total_rows' => 'required|integer|min:1|max:26', // A-Z
        'seats_per_row' => 'required|integer|min:1|max:30',
        'screen_type_id' => 'required|exists:screen_types,id',
        'seat_configs' => 'nullable|array', // Từ preview
    ]);

    // === BƯỚC 2: DATABASE TRANSACTION ===
    DB::beginTransaction();
```

```

try {
    // Tạo phòng
    $room = Room::create([...]);

    // Lấy loại ghế mặc định (Standard)
    $defaultSeatType = SeatType::where('name', 'Standard')->first();

    // === BƯỚC 3: TẠO GHẾ ===
    if (!empty($validated['seat_configs'])) {
        // Trường hợp A: Từ preview (custom config)
        foreach ($validated['seat_configs'] as $config) {
            Seat::create([
                'room_id' => $room->id,
                'seat_row' => $config['row'],
                'seat_number' => $config['number'],
                'seat_code' => $config['row'] . $config['number'], // A1,
                'seat_type_id' => $config['type'],
            ]);
        }
    } else {
        // Trường hợp B: Tự động generate
        $rowLabels = range('A', 'Z');
        for ($row = 0; $row < $validated['total_rows']; $row++) {
            for ($seat = 1; $seat <= $validated['seats_per_row']; $seat++) {
                Seat::create([
                    'room_id' => $room->id,
                    'seat_row' => $rowLabels[$row], // A, B, C...
                    'seat_number' => $seat, // 1, 2, 3...
                    'seat_code' => $rowLabels[$row] . $seat,
                    'seat_type_id' => $defaultSeatType->id,
                ]);
            }
        }
    }

    DB::commit();
    return redirect()->with('success', '...');

} catch (\Exception $e) {
    DB::rollBack();
    return back()->with('error', 'Failed: ' . $e->getMessage());
}

```

### 1.3. Thuật toán tạo sơ đồ ghế (JavaScript)

```

// Trong create.blade.php
function generatePreview() {
    const totalRows = parseInt(totalRowsInput.value);
    const seatsPerRow = parseInt(seatsPerRowInput.value);

```

```

// Khởi tạo seat data với type mặc định = 1 (Standard)
seatData = [];
const rowLabels = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'.split('');

for (let r = 0; r < totalRows; r++) {
  for (let s = 1; s <= seatsPerRow; s++) {
    seatData.push({
      row: rowLabels[r], // A, B, C...
      number: s,         // 1, 2, 3...
      type: 1            // Standard
    });
  }
}

renderSeatMap();
}

```

### Sơ đồ logic render ghế:

THUẬT TOÁN RENDER SEAT MAP	
Input: seatData[] = [{row: 'A', number: 1, type: 1}, ...]	
for (row = 0; row < totalRows; row++) {	
HTML += '<div class="seat-row">'	
HTML += '<div class="row-label">Row ' + rowLabels[row]	
seat = 0	
while (seat < seatsPerRow) {	
currentSeat = seatData[row * seatsPerRow + seat]	
nextSeat = seatData[row * seatsPerRow + seat + 1]	
if (currentSeat.type == 3 &&	
nextSeat?.type == 3 &&	← Couple?
nextSeat?.row == currentSeat.row) {	
// RENDER COUPLE SEAT (rộng gấp đôi)	
HTML += '<button class="seat-btn couple">'	
HTML += currentSeat.number + '-' + nextSeat.number	
HTML += '</button>'	
seat += 2 ← Skip 2 ghế	
} else {	
// RENDER SINGLE SEAT	
HTML += '<button class="seat-btn seat-type-' +	
currentSeat.type + '>'	
HTML += currentSeat.number	
HTML += '</button>'	
seat += 1 ← Skip 1 ghế	

```

    }
  }
  HTML += '</div></div>'
}

```

## 1.4. Các Template tạo sẵn

```

function applyTemplate(template) {
  const totalRows = parseInt(totalRowsInput.value);
  const seatsPerRow = parseInt(seatsPerRowInput.value);

  switch(template) {
    case 'all-standard':
      // Tất cả ghế = Standard (type 1)
      seatData.forEach(seat => seat.type = 1);
      break;

    case 'vip-center':
      // VIP ở giữa phòng
      const vipStartRow = Math.floor(totalRows / 3);
      const vipEndRow = Math.ceil(totalRows * 2 / 3);
      const vipStartSeat = Math.floor(seatsPerRow / 4);
      const vipEndSeat = Math.ceil(seatsPerRow * 3 / 4);

      seatData.forEach(seat => {
        const rowIndex = rowLabels.indexOf(seat.row);
        if (rowIndex >= vipStartRow && rowIndex < vipEndRow &&
            seat.number > vipStartSeat && seat.number <= vipEndSeat) {
          seat.type = 2; // VIP
        }
      });
      break;

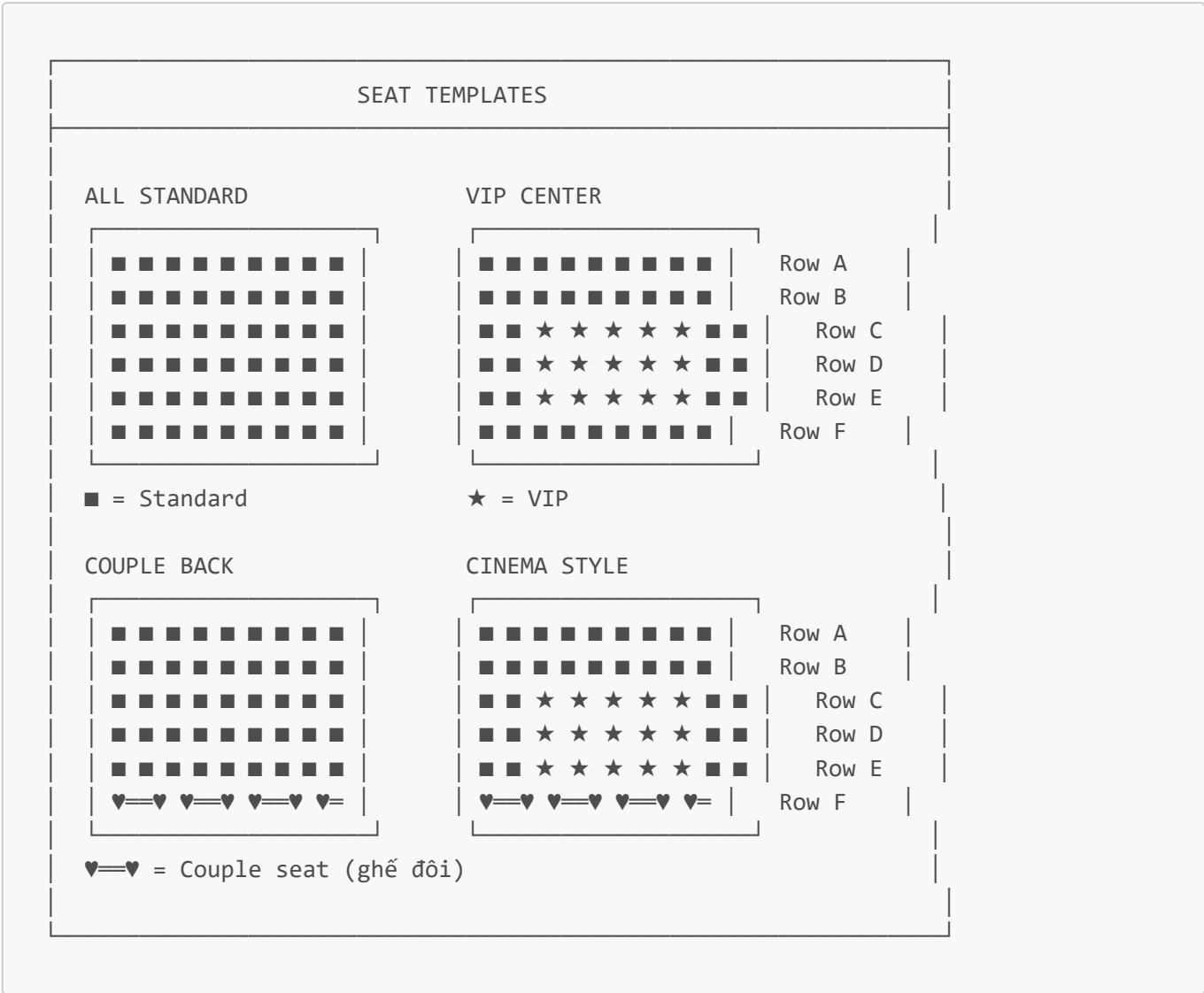
    case 'couple-back':
      // Couple ở hàng cuối
      const lastRowIndex = totalRows - 1;
      seatData.forEach((seat, index) => {
        const rowIndex = rowLabels.indexOf(seat.row);
        if (rowIndex === lastRowIndex && seat.number % 2 === 1) {
          seat.type = 3; // Couple
          if (seatData[index + 1]) {
            seatData[index + 1].type = 3;
          }
        }
      });
      break;

    case 'cinema-style':
      // Phong cách rạp chiếu: Standard trước, VIP giữa, Couple sau

```

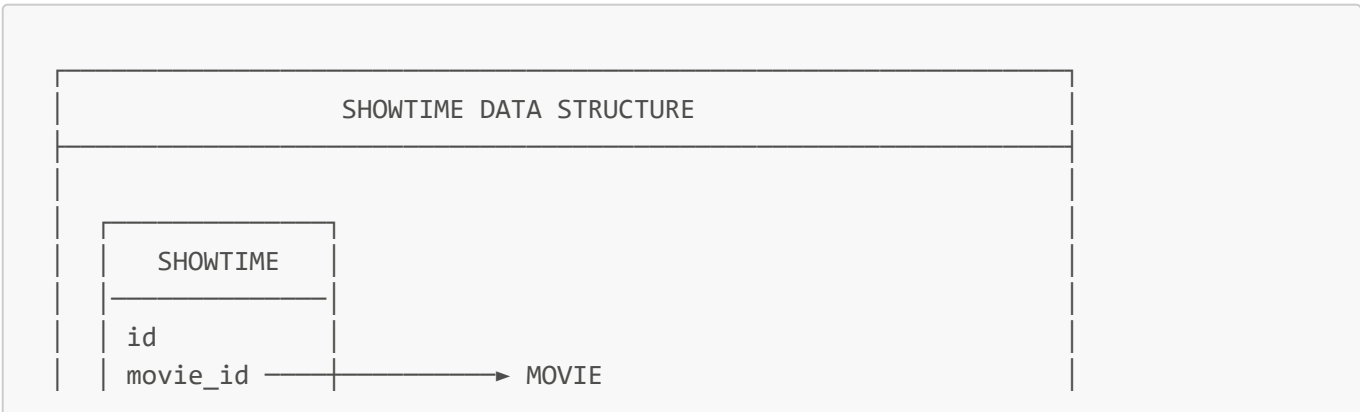
```
// ... kết hợp cả hai template trên
break;
}
```

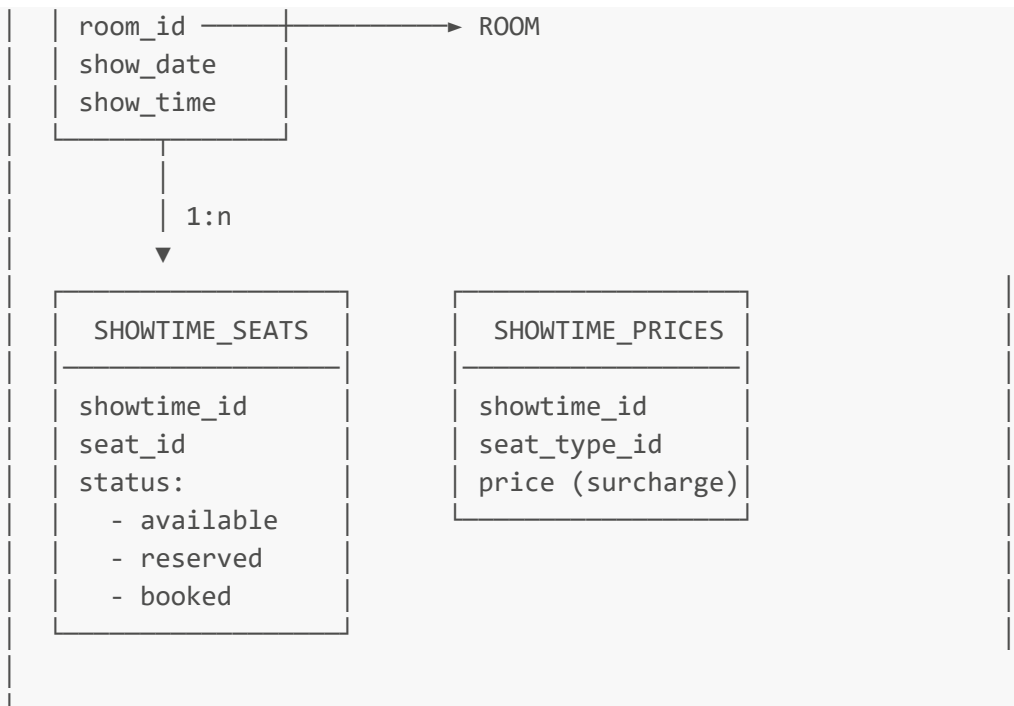
Minh họa các template:



2. SHOWTIME MANAGEMENT - QUẢN LÝ SUẤT CHIẾU

2.1. Cấu trúc dữ liệu Showtime





## 2.2. Thuật toán tạo Showtime

```

public function store(Request $request)
{
    DB::beginTransaction();
    try {
        // === BƯỚC 1: KIỂM TRA TRÙNG LỊCH ===
        $exists = Showtime::where('room_id', $request->room_id)
            ->where('show_date', $request->show_date)
            ->where('show_time', $request->show_time)
            ->exists();

        if ($exists) {
            return back()->with('error', 'This showtime slot is already taken!');
        }

        // === BƯỚC 2: TẠO SHOWTIME ===
        $showtime = Showtime::create([
            'movie_id' => $request->movie_id,
            'room_id' => $request->room_id,
            'show_date' => $request->show_date,
            'show_time' => $request->show_time,
        ]);

        // === BƯỚC 3: TẠO GIÁ CHO TỪNG LOẠI GHẾ ===
        foreach ($request->seat_type_prices as $seatTypeId => $price) {
            ShowtimePrice::create([
                'showtime_id' => $showtime->id,
                'seat_type_id' => $seatTypeId,
                'price' => $price, // Phụ thu giờ cao điểm
            ]);
        }
    }
}
  
```

```

    }

    // === BƯỚC 4: TẠO SHOWTIME_SEATS (TRẠNG THÁI GHẾ) ===
    $room = Room::with('seats')->find($request->room_id);

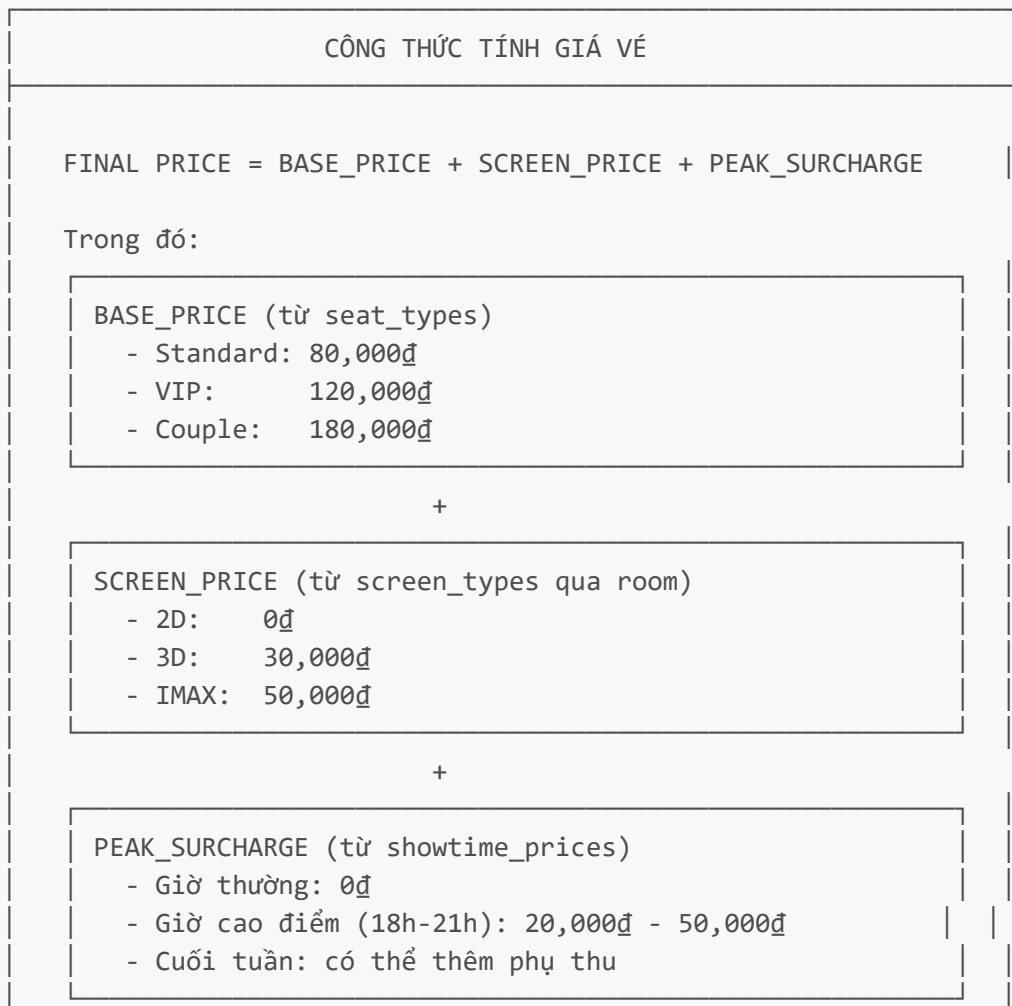
    foreach ($room->seats as $seat) {
        ShowtimeSeat::create([
            'showtime_id' => $showtime->id,
            'seat_id' => $seat->id,
            'status' => 'available', // Ban đầu tất cả đều trống
        ]);
    }

    DB::commit();
    return redirect()->with('success', 'Showtime created!');

} catch (\Exception $e) {
    DB::rollBack();
    return back()->with('error', $e->getMessage());
}
}

```

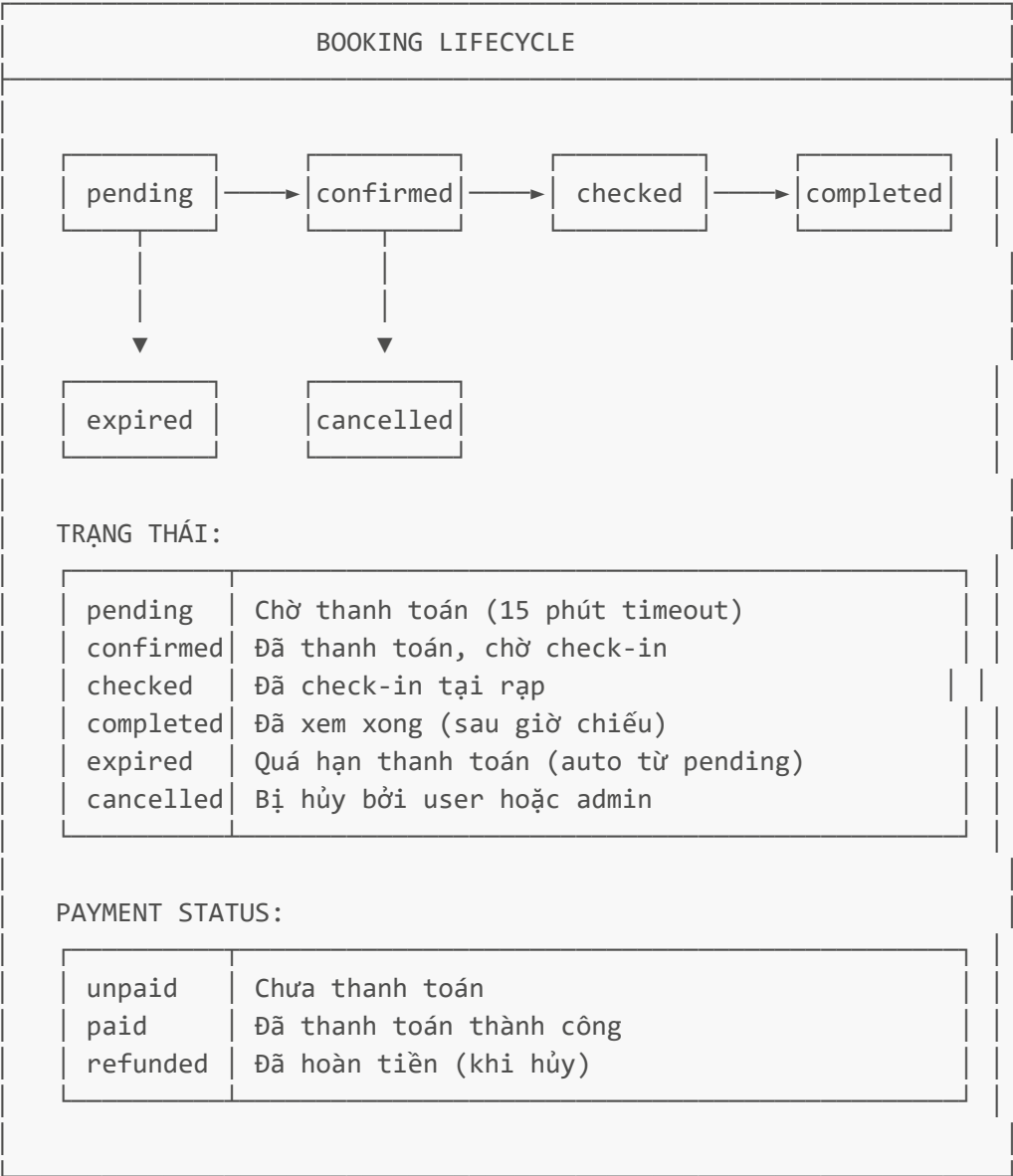
### 2.3. Công thức tính giá vé



VÍ DỤ:  
VIP seat + IMAX room + Peak hour (21:00)  
= 120,000 + 50,000 + 30,000 = 200,000đ

### 3. BOOKING MANAGEMENT - QUẢN LÝ ĐẶT VÉ

#### 3.1. Vòng đời của một Booking



#### 3.2. Thuật toán hủy Booking

```
// AdminBookingController@cancel
public function cancel(Booking $booking)
```

```

{
    // Kiểm tra trạng thái hợp lệ
    if ($booking->status === 'cancelled' || $booking->status === 'expired') {
        return back()->with('error', 'Booking is already cancelled or expired!');
    }

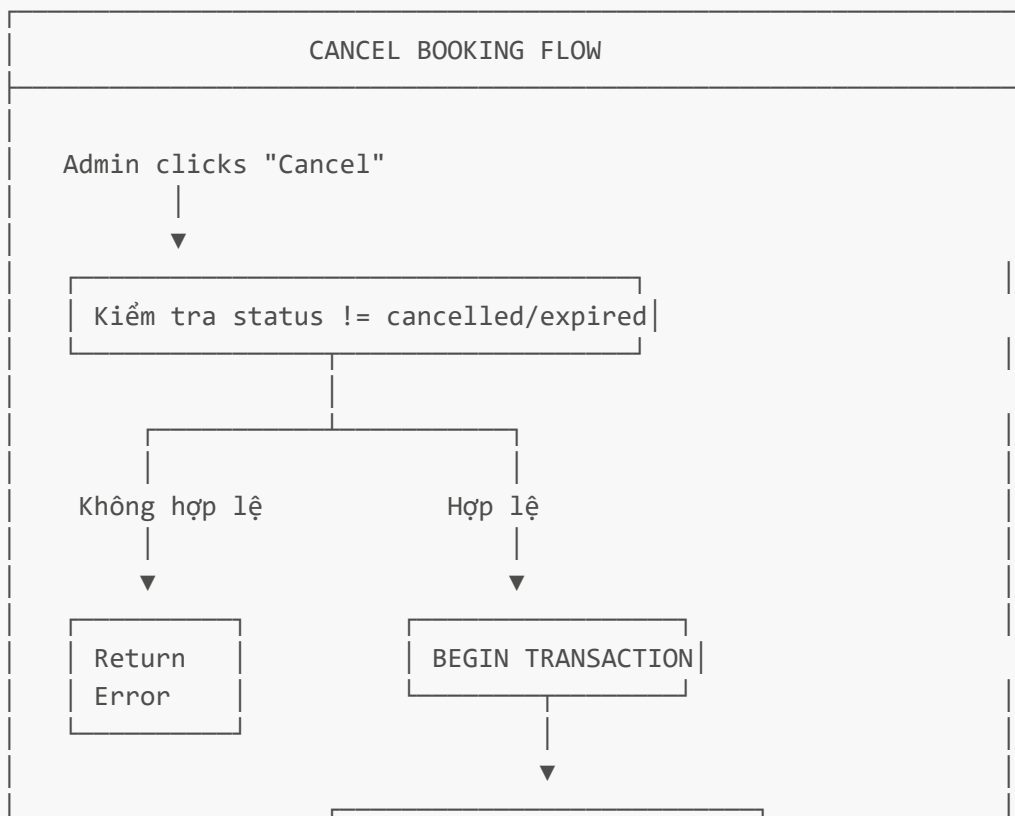
    DB::beginTransaction();
    try {
        // === BƯỚC 1: CẬP NHẬT TRẠNG THÁI ===
        $booking->update(['status' => 'cancelled']);

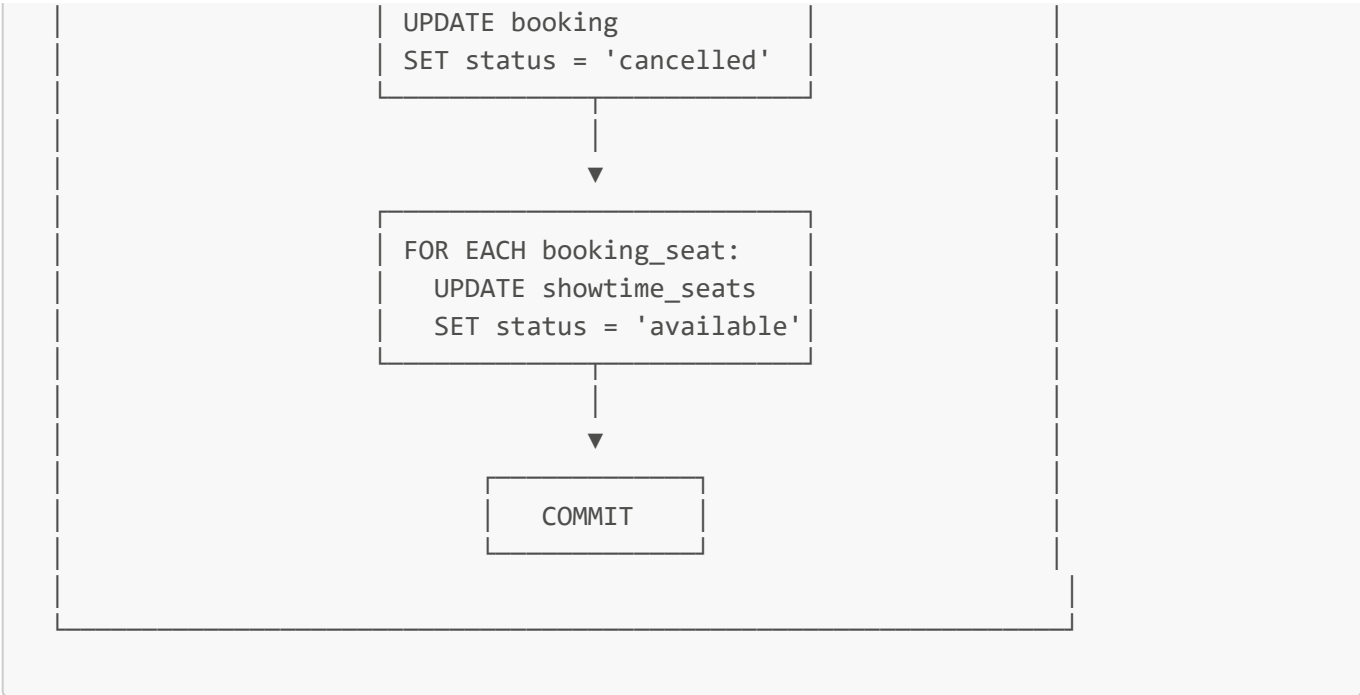
        // === BƯỚC 2: GIẢI PHÓNG GHẾ ===
        foreach ($booking->bookingSeats as $bookingSeat) {
            DB::table('showtime_seats')
                ->where('showtime_id', $booking->showtime_id)
                ->where('seat_id', $bookingSeat->seat_id)
                ->update(['status' => 'available']); // Trả ghế về trống
        }

        DB::commit();
        return back()->with('success', 'Booking cancelled successfully!');
    } catch (\Exception $e) {
        DB::rollBack();
        return back()->with('error', 'Failed: ' . $e->getMessage());
    }
}

```

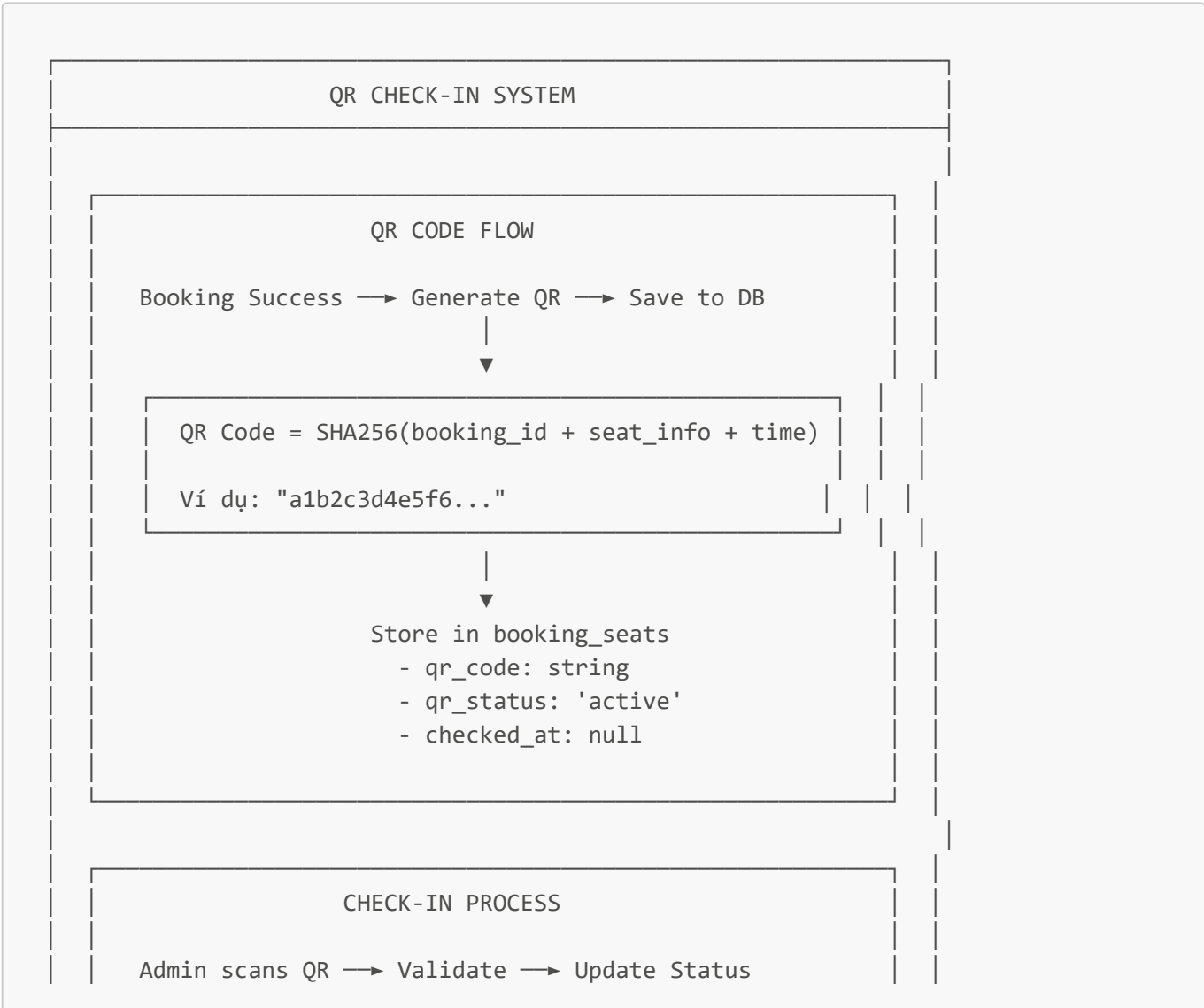
### Sơ đồ luồng hủy booking:





## 4. QR CHECK-IN SYSTEM

### 4.1. Kiến trúc hệ thống QR



Check: qr\_code exists AND qr\_status = 'active'

IF valid:

- qr\_status = 'checked'
- checked\_at = now()
- Return SUCCESS

ELSE:

- Return ERROR (invalid or already used)

## 4.2. Code xử lý Check-in

**Model:** `BookingSeat.php`

```
class BookingSeat extends Model
{
    /**
     * Generate unique QR code
     */
    public static function generateQRCode($bookingId, $seatInfo)
    {
        // SHA256 hash để tạo mã unique
        return hash('sha256', $bookingId . '_' . $seatInfo . '_' .
microtime(true));
    }

    /**
     * Validate QR code
     */
    public static function validateQRCode($qrCode)
    {
        return self::where('qr_code', $qrCode)
            ->where('qr_status', 'active')
            ->exists();
    }

    /**
     * Check-in with QR code
     */
    public static function checkInWithQR($qrCode)
    {
        // Tìm tất cả ghế có cùng QR code (cho couple seat)
        $seats = self::where('qr_code', $qrCode)
```

```

        ->where('qr_status', 'active')
        ->get();

    if ($seats->isEmpty()) {
        return [
            'success' => false,
            'message' => 'QR không hợp lệ hoặc đã được sử dụng'
        ];
    }

    // Cập nhật tất cả ghế có QR code này
    self::where('qr_code', $qrCode)->update([
        'qr_status' => 'checked',
        'checked_at' => now()
    ]);

    return [
        'success' => true,
        'message' => 'Check-in thành công',
        'seats' => $seats->pluck('seat.seat_code'),
        'booking_id' => $seats->first()->booking_id
    ];
}
}

```

### Controller: QRCheckInController.php

```

class QRCheckInController extends Controller
{
    /**
     * Show QR scanner page
     */
    public function index()
    {
        return view('admin.qr_checkin.index');
    }

    /**
     * Preview booking info before check-in
     */
    public function preview(Request $request)
    {
        $qrCode = $request->input('qr_code');

        $bookingSeats = BookingSeat::with([
            'seat',
            'booking.showtime.movie',
            'booking.user'
        ])->where('qr_code', $qrCode)->get();

        if ($bookingSeats->isEmpty()) {
            return response()->json([

```

```

        'success' => false,
        'message' => 'QR code không tồn tại'
    ], 404);
}

$booking = $bookingSeats->first()->booking;

return response()->json([
    'success' => true,
    'data' => [
        'booking_id' => $booking->id,
        'customer_name' => $booking->user->name,
        'movie_title' => $booking->showtime->movie->title,
        'show_date' => $booking->showtime->show_date->format('d/m/Y'),
        'show_time' => $booking->showtime->show_time,
        'seats' => $bookingSeats->pluck('seat.seat_code')->toArray(),
        'qr_status' => $bookingSeats->first()->qr_status,
    ]
]);
}

/**
 * Process check-in
 */
public function checkIn(Request $request)
{
    $result = BookingSeat::checkInWithQR($request->input('qr_code'));

    if (!$result['success']) {
        return response()->json($result, 400);
    }

    // Load thêm thông tin để hiển thị
    $bookingSeats = BookingSeat::with([
        'seat',
        'booking.showtime.movie',
        'booking.user'
    ])->where('qr_code', $request->input('qr_code'))->get();

    $booking = $bookingSeats->first()->booking;

    return response()->json([
        'success' => true,
        'message' => $result['message'],
        'data' => [
            'booking_id' => $booking->id,
            'customer_name' => $booking->user->name,
            'movie_title' => $booking->showtime->movie->title,
            'seats' => $bookingSeats->pluck('seat.seat_code')->toArray(),
            'checked_at' => $bookingSeats->first()->checked_at->format('d/m/Y
H:i:s')
        ]
    ]);
}

```

```
}  
}
```

## 5. DATABASE TRANSACTIONS

### 5.1. Tại sao cần Transaction?

#### VẤN ĐỀ KHI KHÔNG DÙNG TRANSACTION

Kịch bản: Tạo Showtime mới

KHÔNG CÓ TRANSACTION:

```
Step 1: INSERT showtime → SUCCESS ✓  
Step 2: INSERT showtime_prices → SUCCESS ✓  
Step 3: INSERT showtime_seats (100 ghế)  
        → ERROR sau 50 ghế! ✗
```

KẾT QUẢ: Database bị "rác"

- showtime: đã tạo
- showtime\_prices: đã tạo
- showtime\_seats: chỉ có 50/100 ghế

⚠ Data không nhất quán!

CÓ TRANSACTION:

BEGIN TRANSACTION

```
Step 1: INSERT showtime → SUCCESS ✓  
Step 2: INSERT showtime_prices → SUCCESS ✓  
Step 3: INSERT showtime_seats  
        → ERROR sau 50 ghế! ✗
```

ROLLBACK → Tất cả được hoàn tác

KẾT QUẢ: Database vẫn nguyên vẹn

- showtime: KHÔNG có
- showtime\_prices: KHÔNG có
- showtime\_seats: KHÔNG có

✓ Data luôn nhất quán!

### 5.2. Pattern sử dụng Transaction

```
// Pattern chuẩn trong Laravel
DB::beginTransaction();
try {
    // Các thao tác database
    $room = Room::create([...]);
    Seat::create([...]);
    Seat::create([...]);
    // ...

    DB::commit(); // Nếu tất cả OK, lưu vào DB
    return redirect()->with('success', '...');
} catch (\Exception $e) {
    DB::rollBack(); // Nếu có lỗi, hoàn tác tất cả
    return back()->with('error', $e->getMessage());
}
```

---

## 6. FILTER VÀ SEARCH PATTERNS

### 6.1. Query Builder với Filter

**Ví dụ từ AdminUserController:**

```
public function index(Request $request)
{
    // Bắt đầu query
    $query = User::withCount('bookings');

    // Filter by role (nếu có)
    if ($request->role) {
        $query->where('role', $request->role);
    }

    // Search by name hoặc email
    if ($request->search) {
        $search = $request->search;
        $query->where(function($q) use ($search) {
            $q->where('name', 'like', "%{$search}%")
                ->orWhere('email', 'like', "%{$search}%");
        });
    }

    // Sắp xếp và phân trang
    $users = $query->orderBy('created_at', 'desc')->paginate(20);

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

**Giải thích Closure trong where:****CLOSURE TRONG WHERE**

```
$query->where(function($q) use ($search) {
    $q->where('name', 'like', "%{$search}%")
    ->orWhere('email', 'like', "%{$search}%");
});
```

CHUYỂN THÀNH SQL:

```
SELECT * FROM users
WHERE role = 'admin'
AND (
    name LIKE '%john%'
    OR email LIKE '%john%'
)
```

TẠI SAO CẦN CLOSURE?

**✗ KHÔNG DÙNG CLOSURE:**

```
$query->where('role', 'admin')
    ->where('name', 'like', '%john%')
    ->orWhere('email', 'like', '%john%');
```

SQL: role='admin' AND name LIKE '%john%'  
OR email LIKE '%john%'

⚠ orWhere sẽ bỏ qua điều kiện role!

## 7. PAGINATION

### 7.1. Sử dụng Pagination

```
// Trong Controller
$movies = Movie::latest()->paginate(20); // 20 item mỗi trang

// Trong View
@foreach ($movies as $movie)
    {{-- Hiển thị movie --}}
@endforeach

{{-- Hiển thị pagination links --}}
```

```
@if ($movies->hasPages())
<nav>
  <ul class="pagination">
    @foreach ($movies->getUrlRange(1, $movies->lastPage()) as $page => $url)
      <li class="{ $page == $movies->currentPage() ? 'active' : '' }">
        <a href="{ $url }">{ $page }</a>
      </li>
    @endforeach
  </ul>
</nav>
@endif
```

## 7.2. Pagination với Filter (giữ query params)

```
// Pagination tự động giữ lại query params
$movies = Movie::where('status', $request->status)
->paginate(20)
->withQueryString(); // Giữ lại ?status=now_showing&page=2
```

---

## 8. KẾT LUẬN PHẦN 2

### 8.1. Các kỹ thuật quan trọng đã học

Kỹ thuật	Mô tả
<b>Database Transactions</b>	Đảm bảo data integrity khi có nhiều thao tác liên quan
<b>Eager Loading</b>	Tối ưu query, tránh N+1 problem
<b>Query Builder</b>	Xây dựng query linh hoạt với filter và search
<b>Validation</b>	Kiểm tra input trước khi lưu
<b>Flash Messages</b>	Thông báo kết quả thao tác
<b>Pagination</b>	Phân trang hiệu quả cho dữ liệu lớn

### 8.2. Tiếp theo trong Part 3

- Tư duy thiết kế UX/UI cho Admin Panel
- Responsive design patterns
- CSS architecture và naming conventions
- JavaScript interactivity patterns

---

Xem tiếp: **Part 3 - Tư duy UX/UI và thiết kế**