DESIGN THINKING PART 2: TỐI ƯU & PHÁT TRIỂN

"Nếu có thêm thời gian..."

Lời mở đầu

Không có code nào hoàn hảo. Admin Panel hiện tại đã chạy được, nhưng có nhiều điểm có thể cải thiện.

Phần này chia sẻ những gì tôi sẽ làm nếu có thêm thời gian - để bạn có roadmap phát triển tiếp theo.

PHẦN A: TỐI ƯU CODE HIỆN TẠI

1. Refactor Controller - Service Pattern

Vấn đề hiện tại

```
// AdminBookingController.php hiện tại
public function store(Request $request)
{
    // Validate - 10 dòng
    // Calculate price - 15 dòng
    // Check seat availability - 20 dòng
    // Create booking - 10 dòng
    // Create booking seats - 15 dòng
    // Send email - 5 dòng
    // Return response - 5 dòng
}
// Controller quá "béo"
```

Giải pháp: Service Layer

```
// Tạo BookingService.php
class BookingService
{
    public function calculatePrice($showtime, $seats) { }
    public function checkAvailability($showtime, $seats) { }
    public function createBooking($user, $showtime, $seats) { }
    public function sendConfirmation($booking) { }
}

// Controller "gầy" hơn
public function store(Request $request, BookingService $service)
{
    $validated = $request->validated();
    $booking = $service->createBooking(
```

```
            auth()->user(),
            $validated['showtime_id'],
            $validated['seats']
        );
        return redirect()->with('success', 'Booking created');
    }
```

Lợi ích

- **Testable**: Có thể unit test service riêng
- **Reusable**: API và Web dùng chung service
- **Maintainable**: Logic tập trung một chỗ

---

2. Repository Pattern cho Database

Vấn đề hiện tại

```
// Trong controller
$movies = Movie::where('status', 'active')
    ->with('genres')
    ->orderBy('release_date', 'desc')
    ->paginate(15);
```

Logic query nằm rải rác trong controller. Nếu cần thay đổi cách query → sửa nhiều chỗ.

Giải pháp: Repository

```
// MovieRepository.php
interface MovieRepositoryInterface
{
    public function getActiveMovies($perPage = 15);
    public function findWithGenres($id);
    public function getMoviesForDropdown();
}

class MovieRepository implements MovieRepositoryInterface
{
    public function getActiveMovies($perPage = 15)
    {
        return Movie::where('status', 'active')
            ->with('genres')
            ->orderBy('release_date', 'desc')
            ->paginate($perPage);
    }
}

// Controller sử dụng
```

```
public function index(MovieRepositoryInterface $movies)
{
    return view('admin.movies.index', [
        'movies' => $movies->getActiveMovies()
    ]);
}
```

Lợi ích

- **Switchable**: Dễ thay đổi từ MySQL sang MongoDB
- **Mockable**: Dễ test với mock repository
- **Centralized**: Tất cả query ở một chỗ

---

3. Form Request Validation

Vấn đề hiện tại

```
public function store(Request $request)
{
    $validated = $request->validate([
        'title' => 'required|max:255',
        'description' => 'required',
        'duration' => 'required|integer|min:1',
        'release_date' => 'required|date',
        // ... 20 dòng validate rules
    ]);
}
```

Validate rules làm controller dài và khó reuse.

Giải pháp: Form Request

```
// StoreMovieRequest.php
class StoreMovieRequest extends FormRequest
{
    public function rules()
    {
        return [
            'title' => 'required|max:255',
            'description' => 'required',
            'duration' => 'required|integer|min:1',
            'release_date' => 'required|date',
        ];
    }

    public function messages()
    {
```

```
        return [
            'title.required' => 'Tiêu đề phim không được để trống',
            'duration.min' => 'Thời lượng phải lớn hơn 0',
        ];
    }
}

// Controller gọn hơn
public function store(StoreMovieRequest $request)
{
    $movie = Movie::create($request->validated());
    return redirect()->with('success', 'Created');
}
```

Lợi ích

- **Reusable**: Update request có thể extend từ Store
- **Organized**: Validation logic tách riêng
- **Testable**: Có thể test validation độc lập

---

4. Event & Listener Pattern

Vấn đề hiện tại

```
public function confirmBooking($id)
{
    $booking = Booking::find($id);
    $booking->status = 'confirmed';
    $booking->save();

    // Gửi email xác nhận
    Mail::to($booking->user)->send(new BookingConfirmed($booking));

    // Gửi SMS
    SMS::send($booking->user->phone, 'Booking confirmed');

    // Log activity
    Log::info('Booking confirmed', ['id' => $id]);

    // Update analytics
    Analytics::track('booking_confirmed', $booking);
}
```

Một action trigger nhiều side effects. Controller quá tải.

Giải pháp: Events

```php
// BookingConfirmed.php (Event)
class BookingConfirmed
{
    public $booking;
    public function __construct(Booking $booking)
    {
        $this->booking = $booking;
    }
}

// SendConfirmationEmail.php (Listener)
class SendConfirmationEmail
{
    public function handle(BookingConfirmed $event)
    {
        Mail::to($event->booking->user)
            ->send(new BookingConfirmedMail($event->booking));
    }
}

// Controller chỉ cần
public function confirmBooking($id)
{
    $booking = Booking::find($id);
    $booking->update(['status' => 'confirmed']);

    event(new BookingConfirmed($booking));

    return redirect()->with('success', 'Confirmed');
}
```

Lợi ích

- **Decoupled**: Controller không cần biết ai listen
- **Scalable**: Thêm listener mới không sửa controller
- **Async**: Có thể queue listener để chạy background

---

5. Caching Strategy

Vấn đề hiện tại

Mỗi request đều query database, kể cả data ít thay đổi.

Giải pháp: Multi-level Caching

```php
// Cache cho data ít thay đổi
$genres = Cache::remember('all_genres', 3600, function () {
    return Genre::all();
```

```
});

// Cache với tags để invalidate dễ hơn
$movies = Cache::tags(['movies'])->remember('active_movies', 600, function () {
    return Movie::where('status', 'active')->get();
});

// Khi update movie
Cache::tags(['movies'])->flush();
```

Caching Matrix

Data	Cache Time	Invalidation
Genres	24h	Manual
Active Movies	10 min	On CRUD
Dashboard Stats	5 min	Scheduled
User Session	2h	On logout
Seat Availability	30s	Real-time

Lợi ích

- **Performance**: Giảm database load
- **Cost**: Ít query = ít tiền server
- **UX**: Response nhanh hơn

---

6. API Rate Limiting

Vấn đề tiềm ẩn

Admin panel có thể bị spam request (cố tình hoặc vô tình).

Giải pháp

```
// RouteServiceProvider.php
RateLimiter::for('admin', function (Request $request) {
    return Limit::perMinute(60)->by($request->user()->id);
});

// Route
Route::middleware(['auth', 'role:admin', 'throttle:admin'])
    ->group(function () {
        // Admin routes
    });
```

Advanced: Action-based Limiting

```
// Giới hạn action nhạy cảm
RateLimiter::for('delete-action', function (Request $request) {
    return Limit::perHour(10)->by($request->user()->id);
});

// Apply cho route xóa
Route::delete('/movies/{id}', 'destroy')
    ->middleware('throttle:delete-action');
```

PHẦN B: TỐI ƯU DATABASE

1. Index Strategy

Vấn đề hiện tại

Query chậm khi data lớn.

Giải pháp: Đánh index thông minh

```
-- Booking queries thường filter theo
CREATE INDEX idx_bookings_user_status
ON bookings (user_id, status);

-- Showtime queries
CREATE INDEX idx_showtimes_movie_date
ON showtimes (movie_id, show_date, start_time);

-- Seat availability check
CREATE INDEX idx_booking_seats_showtime
ON booking_seats (showtime_id, seat_id, status);
```

Nguyên tắc đánh index

```
1. Column trong WHERE clause
2. Column trong JOIN condition
3. Column trong ORDER BY
4. Column có high cardinality (nhiều giá trị khác nhau)

KHÔNG index:
- Column ít giá trị (boolean, status có 3-4 giá trị)
- Column hay UPDATE
- Table nhỏ (< 1000 rows)
```

2. Query Optimization

Vấn đề: N+1 còn sót

```
// Trang booking list
$bookings = Booking::paginate(20);

// Trong view
@foreach ($bookings as $booking)
    {{ $booking->user->name }}  // N queries
    {{ $booking->showtime->movie->title }} // N queries
    {{ $booking->showtime->room->name }} // N queries
@endforeach
```

Giải pháp: Eager Loading có chọn lọc

```
$bookings = Booking::with([
    'user:id,name,email',  // Chỉ lấy cột cần
    'showtime:id,movie_id,room_id,start_time',
    'showtime.movie:id,title',
    'showtime.room:id,name'
])->paginate(20);
```

Sử dụng Query Builder cho report

```
// Thay vì Eloquent khi cần aggregate
$stats = DB::table('bookings')
    ->select([
        DB::raw('DATE(created_at) as date'),
        DB::raw('COUNT(*) as total'),
        DB::raw('SUM(total_price) as revenue')
    ])
    ->where('status', 'confirmed')
    ->groupBy('date')
    ->get();
```

3. Database Partitioning (Future)

Khi nào cần?

- Table > 10 triệu rows
- Query history data thường xuyên
- Archive old data

Ý tưởng

```
-- Partition bookings theo năm
CREATE TABLE bookings (
    id BIGINT,
    created_at DATETIME,
    ...
) PARTITION BY RANGE (YEAR(created_at)) (
    PARTITION p2023 VALUES LESS THAN (2024),
    PARTITION p2024 VALUES LESS THAN (2025),
    PARTITION p2025 VALUES LESS THAN (2026)
);
```

Query booking 2024 chỉ scan partition p2024.

PHẦN C: TÍNH NĂNG MỚI CHO ADMIN

1. Bulk Actions

Nhu cầu

Admin muốn thực hiện hành động trên nhiều items cùng lúc.

Implementation

```
// Route
Route::post('/admin/movies/bulk-action', 'bulkAction');

// Controller
public function bulkAction(Request $request)
{
    $action = $request->input('action');
    $ids = $request->input('ids');

    switch ($action) {
        case 'delete':
            Movie::whereIn('id', $ids)->delete();
            break;
        case 'activate':
            Movie::whereIn('id', $ids)->update(['status' => 'active']);
            break;
        case 'deactivate':
            Movie::whereIn('id', $ids)->update(['status' => 'inactive']);
            break;
    }

    return back()->with('success', count($ids) . ' items updated');
}
```

UI

```
<table>
    <thead>
        <tr>
            <th><input type="checkbox" id="select-all"></th>
            <th>Title</th>
            ...
        </tr>
    </thead>
    <tbody>
        @foreach($movies as $movie)
        <tr>
            <td><input type="checkbox" name="ids[]" value="{{ $movie->id }}"></td>
            ...
        </tr>
        @endforeach
    </tbody>
</table>

<div class="bulk-actions">
    <select name="action">
        <option value="activate">Activate</option>
        <option value="deactivate">Deactivate</option>
        <option value="delete">Delete</option>
    </select>
    <button type="submit">Apply</button>
</div>
```

---

2. Advanced Filtering

Nhu cầu

Filter phức tạp hơn: date range, multiple status, price range.

Implementation với Query Builder

```
public function index(Request $request)
{
    $query = Booking::query();

    // Date range
    if ($request->filled('date_from')) {
        $query->whereDate('created_at', '>=', $request->date_from);
    }
    if ($request->filled('date_to')) {
        $query->whereDate('created_at', '<=', $request->date_to);
```

```
    }

    // Multiple status
    if ($request->filled('status')) {
        $query->whereIn('status', (array) $request->status);
    }

    // Price range
    if ($request->filled('price_min')) {
        $query->where('total_price', '>=', $request->price_min);
    }
    if ($request->filled('price_max')) {
        $query->where('total_price', '<=', $request->price_max);
    }

    // Search
    if ($request->filled('search')) {
        $query->whereHas('user', function ($q) use ($request) {
            $q->where('name', 'like', '%' . $request->search . '%')
              ->orWhere('email', 'like', '%' . $request->search . '%');
        });
    }

    $bookings = $query->with(['user', 'showtime.movie'])
        ->latest()
        ->paginate(20)
        ->withQueryString();

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

---

3. Export to Excel/PDF

Nhu cầu

Admin cần export data để báo cáo.

Implementation với Laravel Excel

```
// composer require maatwebsite/excel

// BookingsExport.php
class BookingsExport implements FromQuery, WithHeadings, WithMapping
{
    protected $filters;

    public function __construct($filters)
    {
        $this->filters = $filters;
    }
```

```
    public function query()
    {
        return Booking::query()
            ->when($this->filters['status'], fn($q, $status) =>
                $q->where('status', $status)
            )
            ->with(['user', 'showtime.movie']);
    }

    public function headings(): array
    {
        return ['ID', 'User', 'Movie', 'Date', 'Total', 'Status'];
    }

    public function map($booking): array
    {
        return [
            $booking->id,
            $booking->user->name,
            $booking->showtime->movie->title,
            $booking->created_at->format('Y-m-d'),
            $booking->total_price,
            $booking->status,
        ];
    }
}

// Controller
public function export(Request $request)
{
    return Excel::download(
        new BookingsExport($request->all()),
        'bookings-' . now()->format('Y-m-d') . '.xlsx'
    );
}
```

---

4. Activity Log

Nhu cầu

Theo dõi ai làm gì khi nào.

Implementation

```
// Migration
Schema::create('activity_logs', function (Blueprint $table) {
    $table->id();
    $table->foreignId('user_id')->constrained();
    $table->string('action'); // create, update, delete
```

```
    $table->string('model_type'); // App\Models\Movie
    $table->unsignedBigInteger('model_id');
    $table->json('old_values')->nullable();
    $table->json('new_values')->nullable();
    $table->string('ip_address');
    $table->timestamps();
});

// Trait LogsActivity
trait LogsActivity
{
    protected static function bootLogsActivity()
    {
        static::created(function ($model) {
            ActivityLog::create([
                'user_id' => auth()->id(),
                'action' => 'create',
                'model_type' => get_class($model),
                'model_id' => $model->id,
                'new_values' => $model->toArray(),
                'ip_address' => request()->ip(),
            ]);
        });

        static::updated(function ($model) {
            ActivityLog::create([
                'user_id' => auth()->id(),
                'action' => 'update',
                'model_type' => get_class($model),
                'model_id' => $model->id,
                'old_values' => $model->getOriginal(),
                'new_values' => $model->getChanges(),
                'ip_address' => request()->ip(),
            ]);
        });
    }
}

// Sử dụng trong Model
class Movie extends Model
{
    use LogsActivity;
}
```

View Activity Log

```
// Admin có thể xem ai đã làm gì
$logs = ActivityLog::with('user')
    ->where('model_type', Movie::class)
    ->where('model_id', $movieId)
```

```
        ->latest()
        ->get();
```

---

5. Scheduled Tasks

Nhu cầu

- Tự động hủy booking chưa thanh toán sau 15 phút
- Tự động đánh dấu suất chiếu đã qua là completed
- Gửi reminder email trước suất chiếu 2 giờ

Implementation

```
// app/Console/Kernel.php
protected function schedule(Schedule $schedule)
{
    // Hủy booking pending sau 15 phút
    $schedule->call(function () {
        Booking::where('status', 'pending')
            ->where('created_at', '<', now()->subMinutes(15))
            ->update(['status' => 'expired']);
    })->everyMinute();

    // Update showtime status
    $schedule->call(function () {
        Showtime::where('status', 'active')
            ->where('show_date', '<', today())
            ->update(['status' => 'completed']);
    })->dailyAt('00:00');

    // Send reminders
    $schedule->job(new SendShowtimeReminders)->hourly();
}
```

---

6. Real-time Dashboard

Nhu cầu

Dashboard tự động update không cần refresh.

Implementation với Laravel Echo + Pusher

```
// Event
class BookingCreated implements ShouldBroadcast
{
    public $booking;
```

```
    public function broadcastOn()
    {
        return new PrivateChannel('admin-dashboard');
    }
}

// JavaScript
Echo.private('admin-dashboard')
    .listen('BookingCreated', (e) => {
        // Update booking count
        document.getElementById('booking-count').textContent =
            parseInt(document.getElementById('booking-count').textContent) + 1;

        // Update revenue
        document.getElementById('revenue').textContent =
            parseInt(document.getElementById('revenue').textContent) +
e.booking.total_price;

        // Show notification
        showToast('New booking: ' + e.booking.id);
    });
```

PHẦN D: SECURITY ENHANCEMENTS

1. Two-Factor Authentication

```
// Sử dụng Laravel Fortify hoặc custom
public function verify2FA(Request $request)
{
    $user = auth()->user();
    $code = $request->input('code');

    if (Google2FA::verifyKey($user->google2fa_secret, $code)) {
        session(['2fa_verified' => true]);
        return redirect()->intended('/admin');
    }

    return back()->withErrors(['code' => 'Invalid code']);
}
```

2. IP Whitelist

```
// Middleware
class AdminIPWhitelist
{
    protected $whitelist = [
        '192.168.1.0/24',  // Office network
```

```
            '10.0.0.0/8',      // VPN
        ];

        public function handle($request, Closure $next)
        {
            if (!$this->isWhitelisted($request->ip())) {
                abort(403, 'IP not allowed');
            }
            return $next($request);
        }
    }
```

3. Session Security

```
// config/session.php
'secure' => true,           // HTTPS only
'http_only' => true,        // No JavaScript access
'same_site' => 'strict',    // Prevent CSRF

// Regenerate session on login
Auth::login($user);
session()->regenerate();
```

PHẦN E: TESTING STRATEGY

1. Unit Tests

```
// tests/Unit/BookingServiceTest.php
class BookingServiceTest extends TestCase
{
    public function test_calculate_price_with_standard_seats()
    {
        $service = new BookingService();
        $showtime = Showtime::factory()->create(['base_price' => 100000]);
        $seats = Seat::factory()->count(2)->create(['type' => 'standard']);

        $price = $service->calculatePrice($showtime, $seats);

        $this->assertEquals(200000, $price);
    }

    public function test_calculate_price_with_vip_seats()
    {
        // VIP có surcharge 50%
        $service = new BookingService();
        $showtime = Showtime::factory()->create(['base_price' => 100000]);
        $seats = Seat::factory()->count(2)->create(['type' => 'vip']);
```

```
        $price = $service->calculatePrice($showtime, $seats);

        $this->assertEquals(300000, $price); // 100k * 1.5 * 2
    }
}
```

2. Feature Tests

```
// tests/Feature/AdminMovieTest.php
class AdminMovieTest extends TestCase
{
    use RefreshDatabase;

    public function test_admin_can_create_movie()
    {
        $admin = User::factory()->create(['role' => 'admin']);

        $response = $this->actingAs($admin)->post('/admin/movies', [
            'title' => 'Test Movie',
            'description' => 'Description',
            'duration' => 120,
            'release_date' => '2024-01-01',
        ]);

        $response->assertRedirect('/admin/movies');
        $this->assertDatabaseHas('movies', ['title' => 'Test Movie']);
    }

    public function test_non_admin_cannot_access_admin_panel()
    {
        $user = User::factory()->create(['role' => 'user']);

        $response = $this->actingAs($user)->get('/admin/dashboard');

        $response->assertRedirect('/');
    }
}
```

3. Browser Tests

```
// tests/Browser/AdminDashboardTest.php
class AdminDashboardTest extends DuskTestCase
{
    public function test_admin_can_see_dashboard_stats()
    {
        $admin = User::factory()->create(['role' => 'admin']);

        $this->browse(function (Browser $browser) use ($admin) {
            $browser->loginAs($admin)
```

```
                ->visit('/admin/dashboard')
                ->assertSee('Total Revenue')
                ->assertSee('Total Bookings')
                ->assertSee('Total Users');
        });
    }
}
```

PHẦN F: DEPLOYMENT & DEVOPS

1. Environment Configuration

```
# .env.production
APP_ENV=production
APP_DEBUG=false

# Cache & Session
CACHE_DRIVER=redis
SESSION_DRIVER=redis
QUEUE_CONNECTION=redis

# Database
DB_CONNECTION=mysql
DB_HOST=production-db-server
DB_DATABASE=cinebook_prod

# Security
ADMIN_IP_WHITELIST=192.168.1.0/24,10.0.0.0/8
```

2. CI/CD Pipeline

```
# .github/workflows/deploy.yml
name: Deploy

on:
  push:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Run Tests
        run: |
          composer install
          php artisan test
```

```
  deploy:
    needs: test
    runs-on: ubuntu-latest
    steps:
      - name: Deploy to Production
        run: |
          ssh production-server 'cd /var/www/cinebook && git pull && composer
install --no-dev && php artisan migrate --force'
```

3. Monitoring

```
// Sử dụng Laravel Telescope cho development
// Sử dụng Sentry cho production

// config/sentry.php
'dsn' => env('SENTRY_DSN'),

// Trong Exception Handler
public function report(Throwable $exception)
{
    if (app()->bound('sentry') && $this->shouldReport($exception)) {
        app('sentry')->captureException($exception);
    }

    parent::report($exception);
}
```

TÓM TẮT: ROADMAP PHÁT TRIỂN

Phase 1: Code Quality (1-2 tuần)

- ☐ Implement Service Pattern
- ☐ Add Form Request Validation
- ☐ Write Unit Tests cho core logic

Phase 2: Performance (1 tuần)

- ☐ Add Caching Layer
- ☐ Optimize Queries
- ☐ Add Database Indexes

Phase 3: Features (2-3 tuần)

- ☐ Bulk Actions
- ☐ Advanced Filtering
- ☐ Export Excel/PDF
- ☐ Activity Log

Phase 4: Security (1 tuần)

- [ ] Two-Factor Authentication
- [ ] IP Whitelist
- [ ] Security Audit

Phase 5: DevOps (1 tuần)

- [ ] Setup CI/CD
- [ ] Add Monitoring
- [ ] Documentation

Tiếp theo: Phần 3: Business Intelligence - "Kiểm soát lời lỗ của rạp phim"