- Workflow: Search Results → Seats (TrainController@seats) → Passenger Form → DB Booking
  - 1) From search results to seats
  - 2) Controller: loading the seats page
  - 3) Blade: seats UI and selection form
  - 4) Controller: showing the passenger form
  - 5) Blade: passenger form renders one row per seat
  - 6) Saving to DB: controller booking method
  - 7) Database tables and relationships used in this flow
  - 8) End-to-end summary

Workflow: Search Results → Seats (TrainController@seats) → Passenger Form → DB Booking

This document explains how a user moves from the search results into seat selection, then fills passenger details, and finally how the booking is saved to the database. It is written for students with detailed code walk-throughs.

---

1) From search results to seats

- File: `resources/views/trains/search-results.blade.php`
- Each result card includes a "Select Seats" button that links to the seats page with important context params:

```
<a class="btn" href="{{ route('trains.seats', [
    'id' => $t['id'],
    'journey_date' => $searchParams['journey_date'],
    'passengers' => $searchParams['passengers'],
    'route_id' => $t['route_id'] ?? null
]) }}">Select Seats</a>
```

- The query string carries `journey_date`, `passengers`, and `route_id` so the next page knows which date and how many passengers.

Route definition:

```
Route::get('/trains/{id}/seats', [TrainController::class, 'seats'])-
>name('trains.seats');
```

---

2) Controller: loading the seats page

- File: `app/Http/Controllers/TrainController.php`
- Method: `public function seats($trainId)`

Main steps:

1. Load the `Train` by ID.
2. Determine which `Route` to show (either the `route_id` from the query string or first active route):

```
$routeId = request('route_id');
if ($routeId) {
    $route = TrainRoute::with(['fromStation', 'toStation'])
        ->where('train_id', $train->id)
        ->find($routeId);
}
if (!$route) {
    $route = TrainRoute::with(['fromStation', 'toStation'])
        ->where('train_id', $train->id)
        ->where('is_active', true)
        ->orderBy('id')
        ->first();
}
```

3. Determine the start date and build a 7-day strip (today to 6 days ahead or from requested journey date):

```
$today = Carbon::today();
$requested = request('journey_date') ? Carbon::parse(request('journey_date')) :
$today;
$startDate = $requested->lt($today) ? $today : $requested;
```

4. Simulate booked seats per day (demo) with `demoBookedSeats()` to mark the seat grid; shows whether a day is full and how many are available.

5. Prepare a simple `trainData` array for the view.
6. Return Blade: `return view('trains.seats', [...])`.

---

3) Blade: seats UI and selection form

- File: `resources/views/trains/seats.blade.php`
- The page shows:
  - A 7-day date strip where each pill links back to the same route with a different `date` parameter.
  - A legend for Available, Selected, Booked.
  - The actual 4x4 seat grid with labels A1..D4.

Seat grid excerpt:

```
@php
    $seats =
['A1','A2','A3','A4','B1','B2','B3','B4','C1','C2','C3','C4','D1','D2','D3','D4'];
    $rows = array_chunk($seats, 4);
@endphp
@foreach($rows as $row)
  <div class="row">
    @foreach($row as $i => $seat)
      @if($i === 2)
        <div class="aisle">||</div>
      @endif
      @php $taken = in_array($seat, $bookedSeats); @endphp
      <label>
        <input class="seat-checkbox" type="checkbox" name="seats[]" value="{{ $seat
}}" @if($taken) disabled @endif>
        <div class="seat @if($taken) booked @endif">{{ $seat }}</div>
      </label>
    @endforeach
  </div>
@endforeach
```

The form posts to the passenger form route:

```
<form id="seatForm" method="POST" action="{{ route('trains.passengers', ['id' =>
$train['id']]) }}">
  @csrf
  <input type="hidden" name="date" value="{{ $selectedDate }}">
  <input type="hidden" name="passengers" value="{{ request('passengers', 1) }}">
  @if(request('route_id'))
    <input type="hidden" name="route_id" value="{{ request('route_id') }}">
```

```
  @endif
  ...seat checkboxes...
  <button type="submit">Confirm Selection</button>
</form>
```

A small JS ensures at least one seat is selected before submitting.

---

4) Controller: showing the passenger form

- Route:

```
Route::post('/trains/{id}/passengers', [TrainController::class, 'passengerForm'])->name('trains.passengers');
```

- Controller method:

```
public function passengerForm($trainId, Request $request)
{
    $validated = $request->validate([
        'date' => 'required|date',
        'seats' => 'required|array|min:1',
        'seats.*' => 'string',
        'passengers' => 'nullable|integer|min:1|max:4',
        'route_id' => 'nullable|integer',
    ]);

    $train = Train::findOrFail((int)$trainId);
    $route = null;
    if (!empty($validated['route_id'])) {
        $route = TrainRoute::with(['fromStation','toStation'])
            ->where('train_id', $train->id)
            ->find($validated['route_id']);
    }

    // number of passenger rows = #selected seats (bounded by provided passengers if given)
    $count = count($validated['seats']);
    if (!empty($validated['passengers'])) {
        $count = min($count, (int)$validated['passengers']);
    }

    return view('trains.passengers', [
        'train' => $train,
        'route' => $route,
```

```
        'journey_date' => Carbon::parse($validated['date'])->toDateString(),
        'selected_seats' => $validated['seats'],
        'passenger_count' => $count,
    ]);
}
```

---

5) Blade: passenger form renders one row per seat

- File: `resources/views/trains/passengers.blade.php`
- It shows the context (train, route, date, seats) and then renders a number of passenger rows equal to the number of selected seats.
- Important: The form posts directly to the booking creation route.

Excerpt:

```
<form method="POST" action="{{ route('trains.book', ['id' => $train->id]) }}">
  @csrf
  <input type="hidden" name="journey_date" value="{{ $journey_date }}">
  <input type="hidden" name="route_id" value="{{ $route->id ?? '' }}">
  @foreach($selected_seats as $s)
    <input type="hidden" name="seats[]" value="{{ $s }}">
  @endforeach

  @for($i = 0; $i < count($selected_seats); $i++)
    <div class="row">
      <div style="flex:2;">
        <label>Passenger {{ $i + 1 }} Name</label>
        <input type="text" name="passengers[{{ $i }}][name]" required>
      </div>
      <div style="flex:1;">
        <label>Type</label>
        <select name="passengers[{{ $i }}][type]" required>
          <option value="adult">Adult</option>
          <option value="child">Child</option>
        </select>
      </div>
    </div>
  @endfor

  <div style="text-align:right;">
    <button class="btn" type="submit">Save Booking</button>
  </div>
</form>
```

---

6) Saving to DB: controller booking method

- Route:

```
Route::post('/trains/{id}/book', [TrainController::class, 'storeBooking'])->name('trains.book');
```

- Controller method: `storeBooking($trainId, Request $request)`

Steps performed:

1. Validate input:

```
$data = $request->validate([
    'journey_date' => 'required|date',
    'route_id' => 'nullable|integer',
    'seats' => 'required|array|min:1',
    'seats.*' => 'string',
    'passengers' => 'required|array|min:1',
    'passengers.*.name' => 'required|string',
    'passengers.*.type' => 'required|in:adult,child',
]);
```

2. Resolve train and route; if route_id not provided, take the first active route for the train.
3. Compute pricing:

```
$base = $route ? (float)$route->base_price : 0.0;
$total = 0.0;
foreach ($data['passengers'] as $p) {
    $total += $p['type'] === 'child' ? ($base * 0.5) : $base;
}
```

4. Create the `bookings` record:

```
$booking = Booking::create([
    'booking_reference' => 'BR-' . strtoupper(str()->random(6)),
    'train_id' => $train->id,
    'route_id' => $route?->id,
```

```
    'coach_id' => null,
    'journey_date' => Carbon::parse($data['journey_date'])->toDateString(),
    'passenger_name' => $data['passengers'][0]['name'], // primary contact
    'passenger_email' => null,
    'passenger_phone' => null,
    'passenger_count' => count($data['passengers']),
    'total_amount' => $total,
    'booking_status' => 'pending',
    'payment_status' => 'unpaid',
]);
```

5. Create rows in `booking_seats` for each passenger with seat mapping:

```
$seatModels = Seat::whereIn('seat_number', $data['seats'])->get()-
>keyBy('seat_number');

foreach ($data['passengers'] as $i => $p) {
    $label = $data['seats'][$i] ?? null;
    $seatId = $label && isset($seatModels[$label]) ? $seatModels[$label]->id :
null;

    BookingSeat::create([
        'booking_id' => $booking->id,
        'seat_id' => $seatId,
        'passenger_name' => $p['name'],
        'passenger_age' => null,
        'passenger_gender' => null,
    ]);
}
```

6. Redirect to payment page with the created booking ID:

```
return redirect()->route('payment.create', ['booking' => $booking->id])
    ->with('success', 'Booking created. Proceed to payment.');
```

---

7) Database tables and relationships used in this flow

- `trains` (Train model)
- `routes` (Route model) — `base_price`, `is_active`, `from_station_id`, `to_station_id`
- `seats` (Seat model) — to map seat label (e.g., A1) to `seat_id` if present

- `bookings` (Booking model) — master record storing total, status, and journey date
- `booking_seats` (BookingSeat model) — details per passenger and seat

Key relationships:

- Booking hasMany BookingSeat (`Booking::bookingSeats()`)
- Booking belongsTo Train and Route
- BookingSeat belongsTo Booking and Seat

---

8) End-to-end summary

1. From search results, the user clicks Select Seats with date/passengers/route preserved.
2. Seats page renders a 4x4 grid, marks booked seats (demo), and lets the user choose seats.
3. On submit, server validates and shows a passenger form with one row per selected seat.
4. User fills names and adult/child types.
5. On submit, controller creates a Booking with computed total and creates BookingSeat rows per passenger.
6. User is redirected to payment.

This demonstrates MVC in action: controller orchestrates, views collect/confirm user input, models interact with DB, and relationships keep the data consistent.