- Workflow: Controller (TrainController@seats + @passengerForm + @storeBooking) + seats.blade.php + Database
  - 1) Entry point: from Search Results to Seats
  - 2) Controller: TrainController@seats
    - 2.1 Load Train and Route
    - 2.2 Build date window
    - 2.3 Build week array (availability overview)
    - 2.4 Clamp selected date and compute booked seats
    - 2.5 Prepare data for the view
  - 3) View: resources/views/trains/seats.blade.php
    - 3.1 Posting to passenger form
    - 3.2 Seat grid pattern
  - 4) Controller: TrainController@passengerForm
  - 5) View: resources/views/trains/passengers.blade.php
  - 6) Controller: TrainController@storeBooking
    - 6.1 Database writes
  - 7) Data model recap used here
  - 8) Full sequence diagram (text)

Workflow: Controller (TrainController@seats + @passengerForm + @storeBooking) + seats.blade.php + Database

This document explains the complete flow from selecting seats to submitting passenger information and saving the booking (with total) into the database. It's step - by - step for students, with code references from this project.

---

1) Entry point: from Search Results to Seats

On the search results page, the user clicks "Select Seats". That link contains:

- train id
- journey_date
- passengers (count)
- route_id

It routes to `trains.seats` → `TrainController@seats`.

Route (routes/web.php):

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

---

2) Controller: TrainController@seats

File: app/Http/Controllers/TrainController.php

High - level responsibilities:

- Load the Train and (optionally) the requested Route.
- Build a 7 - day strip starting from max(today, journey_date).
- For each day, compute a demo booked seat set (this app uses a fixed demo function for clarity).
- Render the seat grid, mark booked seats disabled.

2.1 Load Train and Route

```
$train = Train::find((int) $trainId);
if (!$train) abort(404, 'Train not found');

$route = null;
$routeId = request('route_id');
if ($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();
}
```

2.2 Build date window

```
$allSeats =
['A1','A2','A3','A4','B1','B2','B3','B4','C1','C2','C3','C4','D1','D2','D3','D4'];
$today = Carbon::today();
$requested = request('journey_date') ? Carbon::parse(request('journey_date')) :
$today;
$startDate = $requested->lt($today) ? $today : $requested;

$selectedDate = request('date') ?: $startDate->toDateString();
$selected = Carbon::parse($selectedDate);
```

2.3 Build week array (availability overview)

```
$week = [];
for ($i = 0; $i < 7; $i++) {
    $d = $startDate->copy()->addDays($i);
    $bookedForDay = $this->demoBookedSeats($d, $allSeats);
    $week[] = [
        'date' => $d->toDateString(),
        'label' => $d->format('D d M'),
        'booked_count' => count($bookedForDay),
        'is_full' => count($bookedForDay) === count($allSeats),
        'available' => count($allSeats) - count($bookedForDay),
    ];
}
```

2.4 Clamp selected date and compute booked seats

```
if ($selected->lt($startDate) || $selected->gt($startDate->copy()->addDays(6))) {
    $selected = $startDate->copy();
    $selectedDate = $selected->toDateString();
}
```

```
$bookedSeats = $this->demoBookedSeats($selected, $allSeats);
$isFullyBooked = count($bookedSeats) === count($allSeats);
```

2.5 Prepare data for the view

```
$trainData = [
    'id' => $train->id,
    'name' => $train->name,
    'number' => $train->number,
    'from' => $route && $route->fromStation ? $route->fromStation->name : '-',
    'to' => $route && $route->toStation ? $route->toStation->name : '-',
    'total_seats' => count($allSeats),
    'available_seats' => count($allSeats) - count($bookedSeats),
    'journey_date' => $startDate->toDateString(),
];

return view('trains.seats', [
    'train' => $trainData,
    'week' => $week,
    'selectedDate' => $selectedDate,
    'bookedSeats' => $bookedSeats,
    'isFullyBooked' => $isFullyBooked,
]);
```

---

3) View: resources/views/trains/seats.blade.php

Main responsibilities:

- Display a 7 - day strip with availability.
- Show a 4x4 seat grid (A..D rows, 1..4 columns) with an aisle.
- Disable seats that are booked for the selected day.
- Post the selected seats with date, passengers, route_id to the passenger form route.

3.1 Posting to passenger form

```
<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
  ... seats checkboxes ...
</form>
```

- At least one seat must be checked (JS enforces it).

3.2 Seat grid pattern

```
@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
```

---

4) Controller: TrainController@passengerForm

When the seat form is submitted, it posts to:

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

The action validates and renders the passenger form with a number of rows equal to selected seats.

```
$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']);
}

$count = count($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,
]);
```

Key point: the form count equals the number of selected seats (capped by provided passengers count if present).

---

5) View: resources/views/trains/passengers.blade.php

- Renders passenger rows equal to the number of selected seats.
- Submits directly to the booking creation endpoint.

```
<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) Controller: TrainController@storeBooking

This saves the booking and its passenger-seat rows, and redirects to payment.

Route:

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

Action:

```
$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',
  ]);

  $train = Train::findOrFail((int)$trainId);
  $route = !empty($data['route_id'])
    ? TrainRoute::where('train_id', $train->id)->find($data['route_id'])
    : TrainRoute::where('train_id', $train->id)->where('is_active', true)->first();

  $base = $route ? (float)$route->base_price : 0.0;  // adult fare
  $total = 0.0;
  foreach ($data['passengers'] as $p) {
    $total += $p['type'] === 'child' ? ($base * 0.5) : $base; // child 50%
  }

  $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',
  ]);

  $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,
    ]);
  }

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

6.1 Database writes

- `bookings` row (total_amount holds the computed total).
- Multiple `booking_seats` rows (one per passenger/seat) linking to the new booking.

---

7) Data model recap used here

- `Train` ↔ `Route` (one-to-many) for schedule and price
- `Coach` (optional here; used earlier for availability math)
- `Seat` (maps seat_number like A1 → seat id)
- `Booking` (stores overall purchase/total and statuses)
- `BookingSeat` (stores passenger name and seat_id for that booking)

---

8) Full sequence diagram (text)

Search Results → Seats link (GET /trains/{id}/seats?journey_date=...&route_id=...) → TrainController@seats builds the grid and booked seats → seats.blade renders checkboxes → User selects seats → POST to /trains/{id}/passengers → TrainController@passengerForm validates, derives count from selected seats, returns passengers.blade → User fills names/types → POST to /trains/{id}/book → TrainController@storeBooking validates, computes total, creates Booking + BookingSeat, redirects to payment

That’s the end - to - end seat selection to database save workflow.