# **Online Bus Ticket Booking Application Documentation**

## **1. Overview**

The **Online Bus Ticket Booking Application** is a full‑stack application designed to let users search for bus routes, book tickets, enter passenger details, and view or download booking confirmations. The application uses Spring Boot with MongoDB on the backend and HTML, CSS, and JavaScript on the frontend. Its modern design is inspired by leading travel portals like RedBus and MakeMyTrip.

## **2. Architecture**

### **2.1 Backend**

* **Framework:** Spring Boot (v2.7.6)
* **Database:** MongoDB (hosted on MongoDB Atlas)
* **Entities:**
  + **Bus:** Stores bus details such as bus number, route (from/to), departure time, arrival time, price, available seats, and seat map.
  + **Booking:** Holds booking information including passenger ID, bus ID, selected seat numbers, booking date, boarding/drop points, booking status, and embedded bus details (bus number, departure time, arrival time) plus passenger details.
  + **Passenger:** Contains passenger information (name, email, contact number, password).
  + **PassengerDetail:** Represents each passenger's details for multi‑seat bookings (name, age, gender).
  + **Seat:** Represents an individual seat (seat ID and status).
* **REST API Controllers:**
  + **PassengerController:** Manages user registration and authentication.
  + **BusController:** Handles creation and search of bus routes.
  + **BookingController:** Manages booking creation (updates seat statuses and embeds bus details) and booking retrieval.
* **Global Configuration:**
  + **CORS Configuration:** Allows cross-origin requests.
  + **Global Exception Handling:** Provides uniform error responses.

### **2.2 Frontend**

* **Technologies:** HTML, CSS, JavaScript, and Bootstrap 5.
* **Pages:**
  + **index.html:** Homepage with navigation links.
  + **passenger\_login.html:** Login page for passengers.
  + **registration.html:** Passenger registration page.
  + **passenger\_dashboard.html:** Dashboard showing booking history and navigation.
  + **search\_bus.html & bus\_routes.html:** Pages to search and display bus routes.
  + **booking\_form.html:** Interactive seat map for selecting seats and entering boarding/drop points.
  + **passenger\_details.html:** Form for entering details of each passenger (if multiple seats booked).
  + **payment.html:** Payment page (with a default OTP of 1111 for simulation).
  + **booking\_confirmation.html:** Confirmation page after successful booking.
  + **booking\_history.html:** Displays past bookings with options to view and download ticket details.
  + **view\_booking.html:** Detailed view of a booking, including bus and passenger details.
  + **user\_profile.html:** (Optional) Allows users to view and update profile information.
* **JavaScript:** A centralized **main.js** file manages functionality such as API calls (with a configurable API\_BASE\_URL), seat selection, booking flow, and PDF generation (using jsPDF).
* **CSS:** Custom styling in **styles.css** complements Bootstrap for a polished, professional look.

## **3. Setup and Configuration**

### **3.1 Backend Setup**

1. **Clone the Repository:** Clone the GitHub repository containing the source code.

**Configure MongoDB:**  Update application.properties (located in src/main/resources) with your MongoDB connection string:

spring.data.mongodb.uri=mongodb+srv://user:Priya123@clusterdeployment.kirge.mongodb.net/bus\_ticket\_db?retryWrites=true&w=majority&appName=ClusterDeployment

server.port=8080

**Build and Run:**  Use Maven to build and run the application:  
 mvn clean install

mvn spring-boot:run

1. **Swagger UI (Optional):** Access API documentation via Swagger UI at:  
    http://localhost:8080/swagger-ui.html

### **3.2 Frontend Setup**

1. **Static Files Location:** Place all HTML, CSS, JavaScript, and images under src/main/webapp.
2. **Serve the Frontend:** Serve these files using a local static server (e.g., VS Code Live Server) or configure Spring Boot to serve static resources (e.g., by placing them in /static).

**API Base URL:** In js/main.js, verify that the API\_BASE\_URL is set correctly:  
  
 const API\_BASE\_URL = "http://localhost:8080/api";

## 

## 

## 

## 

## **4. API Endpoints Overview**

### **4.1 Passenger Endpoints**

**POST /api/passengers/register** Registers a new passenger.

*Request Body:*  
{

"name": "Alice Smith",

"email": "alice@example.com",

"contactNumber": "9876543210",

"password": "alice123"

}

* *Response:* Passenger object with generated ID or 409 Conflict if email already exists.

**POST /api/passengers/login** Authenticates a passenger.  
 *Request Body:* {

"email": "alice@example.com",

"password": "alice123"

}

* *Response:* Passenger object if credentials are valid; otherwise, 401 Unauthorized.

### **4.2 Bus Endpoints**

* **POST /api/buses** Creates a new bus with schedule and seat map.
* **GET /api/buses/search?from={from}&to={to}** Returns buses that match the source and destination.
* **GET /api/buses/{id}** Retrieves details of a specific bus.

### 

### **4.3 Booking Endpoints**

**POST /api/bookings** Creates a new booking. This endpoint updates seat statuses and embeds bus details into the booking record.  
 *Request Body Example:*  
{

"passengerId": "passenger\_id\_here",

"busId": "bus\_id\_here",

"seatNumbers": ["A1", "B1"],

"bookingDate": "2025-05-01T09:00:00",

"status": "CONFIRMED",

"boardingPoint": "Main Bus Stand, Bangalore",

"dropPoint": "Chennai Central"

}

* **GET /api/bookings/passenger/{passengerId}** Retrieves all bookings for a specific passenger.
* **GET /api/bookings/{bookingId}** Retrieves detailed information for a single booking.

## **5. Frontend Flow**

1. **User Registration & Login:**
   * Passengers register via registration.html and log in via passenger\_login.html.
   * On successful login, the Passenger ID is stored in localStorage for use in the booking flow.
2. **Bus Search and Route Display:**
   * Users search for routes on search\_bus.html, and results are shown on bus\_routes.html.
3. **Booking Process:**
   * **Booking Form:** On booking\_form.html, users select seats from an interactive seat map and enter boarding and drop points.
   * **Passenger Details:** For multi‑seat bookings, passenger\_details.html collects details (name, age, gender) for each seat selected.
   * **Payment:** payment.html simulates payment using a default OTP (1111).
   * **Confirmation:** After payment, users are redirected to booking\_confirmation.html.
4. **Booking History & Ticket Viewing:**
   * booking\_history.html displays a list of past bookings with options to view detailed booking information (view\_booking.html) and download the ticket as a PDF.
5. **User Profile (Optional):**
   * user\_profile.html allows users to view and update their personal information.

## **6. Testing and Quality Assurance**

### **6.1 Integration Tests**

Integration tests are written using JUnit 5 and Spring Boot Test with MockMvc. The test suite (ApplicationIntegrationTest.java) covers:

* Passenger registration and login.
* Bus insertion and search.
* Booking creation.
* Booking history retrieval.

To run the tests, execute:

mvn test

### **6.2 Manual Testing**

1. **Launch the Backend:** Run mvn spring-boot:run to start the backend.
2. **Open the Frontend:** Access the frontend pages via your local static file server or Spring Boot’s default static content serving.
3. **Test the Flow:**
   * Register and log in as a passenger.
   * Search for bus routes.
   * Complete the booking process (select seats, enter passenger details, complete payment).
   * Verify booking history, view detailed booking information, and download the ticket as a PDF.

## **7. Deployment and Submission**

1. **GitHub Repository:** Push all project files to GitHub.
2. **Deployable URL:**

Either deploy the application (e.g., on Heroku, AWS)

## **8. Conclusion**

This documentation provides a complete guide for the Online Bus Ticket Booking Application—from system architecture and API endpoints to frontend flow, testing, and deployment instructions. Use this guide to set up, configure, and run the project. For any issues or further modifications, please consult the source code and project logs or reach out for support.