RS-Express Bus Booking System - Technical Documentation

File Structure Analysis

Backend Structure (`Back-end(working)`)

Root Files

- `artisan` - Laravel command-line interface
- `composer.json` - PHP dependency management
- `composer.lock` - Locked PHP dependencies
- `phpunit.xml` - Testing configuration
- `vite.config.ts` - Build tool configuration
- `.env` - Environment configuration (not tracked in git)

Directories

App Directory (`app/`)

Controllers (`app/Http/Controllers/`)

- `AuthController.php` - Handles authentication (login, signup, user data)
- `BookingController.php` - Manages bus bookings
- `BusRegisterController.php` - Controls bus registration
- `BusRouteController.php` - Manages bus routes
- `BusTripController.php` - Handles trip scheduling
- `CancellationController.php` - Processes booking cancellations
- `DashboardController.php` - Admin dashboard data
- `LoyaltyMemberController.php` - Loyalty program management
- `UserController.php` - User management

Models (`app/Models/`)

- `Booking.php` - Booking data model and relationships
  - Relationships: belongsTo User, belongsTo BusTrip
  - Key methods: scopeConfirmed(), getTotalRevenue()
- `BusRegister.php` - Bus registration model
  - Relationships: hasMany BusTrip
  - Properties: bus_no, bus_type, seat_capacity, status
- `BusRoute.php` - Bus route information
  - Relationships: hasMany BusTrip
  - Properties: route_name, start_point, end_point, distance
- `BusTrip.php` - Trip scheduling model
  - Relationships: belongsTo BusRegister, belongsTo BusRoute, hasMany Booking
  - Properties: departure_time, arrival_time, price, available_seats

- `Cancellation.php` - Cancelled booking records
  - Relationships: belongsTo User, belongsTo BusTrip
  - Properties: reason, refund_amount, cancellation_date
- `GuestBooking.php` - Non-user booking records
  - Properties: name, email, phone, similar to Booking
- `LoyaltyMember.php` - Loyalty program membership
  - Relationships: belongsTo User
  - Methods: createForUser(), addPoints(), calculateTier()
- `User.php` - User account model
  - Relationships: hasMany Booking, hasOne LoyaltyMember
  - Properties: name, email, role (admin, staff, agent, user)

Middleware (`app/Http/Middleware/`)

- `Authenticate.php` - Authentication verification
- `CheckRole.php` - Role-based access control
- `VerifyCsrfToken.php` - CSRF protection

Services (`app/Services/`)

- `BookingService.php` - Booking business logic
- `BusService.php` - Bus management logic
- `LoyaltyService.php` - Loyalty program calculations
- `PaymentService.php` - Payment processing

Database Directory (`database/`)

Migrations (`database/migrations/`)

- `2014_10_12_000000_create_users_table.php`
- `2023_01_15_create_bus_register_table.php`
- `2023_01_15_create_bus_routes_table.php`
- `2023_01_15_create_bus_trips_table.php`
- `2023_01_15_create_bookings_table.php`
- `2023_01_15_create_guest_bookings_table.php`
- `2023_01_15_create_cancellations_table.php`
- `2023_01_15_create_loyalty_members_table.php`

Seeders (`database/seeders/`)

- `DatabaseSeeder.php` - Main seeder that calls others
- `UserSeeder.php` - Creates default users
- `BusRouteSeeder.php` - Creates common bus routes
- `SampleBusTripSeeder.php` - Creates sample bus trips
- `DashboardTestDataSeeder.php` - Creates test data for dashboard

Route Files (`routes/`)

- `api.php` - API endpoints
- `admin_menu_api.php` - Admin panel API routes
- `auth.php` - Authentication routes
- `web.php` - Web routes

Configuration Files (`config/`)

- `app.php` - Application configuration
- `auth.php` - Authentication configuration
- `cors.php` - Cross-Origin Resource Sharing settings
- `sanctum.php` - API token authentication
- `database.php` - Database connection settings

Frontend Structure (`bus(working)`)

Root Files

- `package.json` - JavaScript dependency management
- `tailwind.config.js` - CSS framework configuration
- `index.html` - Root HTML template

Directories

Source Directory (`src/`)

Core Files

- `App.js` - Root component and route definitions
- `AppWithPermissions.js` - Permissions wrapper for App
- `index.js` - Application entry point

Pages (`src/pages/`)

- `Homepage/HomePage.js` - Landing page with search
- `Login.js` - User authentication
- `SignUp.js` - User registration
- `AgentPanel.js` - Agent dashboard
- `CompleteProfile.js` - Profile completion form

Components (`src/components/`)

- `Navbar.js` - Navigation header
  - Imported by: App.js
  - Imports: UserProfileDropdown.js, ResponsiveMenu.js
- `UserProfileDropdown.js` - User menu
- `BusCard/BusCard.js` - Bus search result item
- `BusCard/BusList.js` - List of buses
- `SeatBooking/SeatBooking.js` - Seat selection interface
- `Auth/GoogleAuthButton.js` - Google OAuth button
- `Other/PassengerDashboard.js` - User dashboard
- `Other/ResponsiveMenu.js` - Mobile navigation

Admin Module (`src/admin/`)

- `AdminRoutes.js` - Admin panel routing
  - Imports all admin pages
  - Protected by permissions
- `pages/DashboardPage.js` - Admin dashboard
- `pages/BusSchedulePage.js` - Bus schedule management
- `pages/BusRouteManagementPage.js` - Route management
- `pages/BusRegisterPage.js` - Bus registration
- `pages/booking/BusBookingPage.js` - Booking management
- `pages/booking/FreezingSeatPage.js` - Seat reservation
- `pages/payment/OnlinePaymentPage.js` - Online payments
- `pages/payment/AgentPaymentPage.js` - Agent payments
- `pages/LoyaltyMembersPage.jsx` - Loyalty program management

Context (`src/context/`)

- `AuthContext.js` - Authentication state management
  - Used by: App.js, Login.js, Navbar.js, etc.
  - Provides: user, setUser, token, setToken
- `PermissionsContext.js` - User permissions management
  - Used by: AdminRoutes.js, Login.js, Navbar.js
  - Provides: permissions, loadPermissions
- `PrivateRoute.js` - Route protection component
  - Used by: App.js for protected routes

Services (`src/services/`)

- `authService.js` - Authentication API calls
  - Methods: loginUser, fetchUser, signupUser
  - Used by: Login.js, SignUp.js
- `userService.js` - User management API calls
- `busService.js` - Bus data API calls
- `bookingService.js` - Booking API calls
- `loyaltyService.js` - Loyalty program API calls
- `staffService.js` - Staff management API calls

Utilities (`src/utils/`)

- `auth.js` - Token management
  - Methods: getToken, setToken, removeToken
- `axiosConfig.js` - HTTP client configuration
  - Sets up Axios with CSRF handling
- `date.js` - Date formatting utilities
- `permissionUtils.js` - Permission checking helpers

Assets (`src/assets/`)

- `Side.png` - Logo image
- `BusImage.jpg` - Bus imagery
- Various icons and images

Key Connection Points

Authentication Flow

1. `Login.js` calls `loginUser()` from `authService.js`
2. On success, token stored in localStorage via `auth.js`
3. `AuthContext.js` updated with user data
4. `PermissionsContext.js` loads permissions for user
5. Redirect based on permissions to admin or user dashboard

Bus Search & Booking Flow

1. `HomePage.js` search form sends parameters to `BusList.js`
2. `BusList.js` calls `busService.js` to fetch available buses
3. Results displayed as `BusCard.js` components
4. User selects bus, navigates to `SeatBooking.js`
5. `SeatBooking.js` calls `bookingService.js` to create booking

Admin Panel Access Control

1. User logs in via `Login.js`
2. `PermissionsContext.js` loads role-specific permissions
3. `AdminRoutes.js` checks permissions before rendering admin pages
4. `Navbar.js` shows/hides admin panel link based on permissions

Data Flow Between Components

1. **Global State**: Managed by Context providers
   - `AuthContext.js` - User authentication state
   - `PermissionsContext.js` - User permissions
2. **API Communication**: Handled by service modules
   - All API calls use Axios with configuration from `axiosConfig.js`
   - Token automatically included in requests via interceptors
3. **Props Passing**: For parent-child communication
   - E.g., `BusList.js` passes bus data to `BusCard.js`

Database Relationships

1. **User - Booking**: One-to-many
   - User has many bookings
   - Defined in `User.php` and `Booking.php`
2. **BusRegister - BusTrip**: One-to-many
   - Each registered bus has many trips
   - Defined in `BusRegister.php` and `BusTrip.php`
3. **BusRoute - BusTrip**: One-to-many
   - Each route has many trips
   - Defined in `BusRoute.php` and `BusTrip.php`
4. **BusTrip - Booking**: One-to-many
   - Each trip has many bookings
   - Defined in `BusTrip.php` and `Booking.php`
5. **User - LoyaltyMember**: One-to-one
   - User can have one loyalty membership
   - Defined in `User.php` and `LoyaltyMember.php`

Total File Count

- **Backend Files**: ~120 files
    - Controllers: 10
    - Models: 8
    - Migrations: 12
    - Configuration: 15
    - Routes: 5
    - Services: 4
    - Middleware: 6
    - Other: 60+
- **Frontend Files**: ~85 files
    - Components: 30
    - Pages: 12
    - Admin Pages: 15
    - Services: 7
    - Context: 4
    - Utilities: 8
    - Assets: 9+

Technical Challenges and Solutions

1. CSRF Token Management

Challenge: CSRF token mismatches when moving project between environments. **Solution**: Implemented in `axiosConfig.js` with:

- `withCredentials: true` for cross-domain cookie handling
- Interceptor to extract and include CSRF token in requests
- Sanctum configuration in backend for domain whitelisting

2. Role-Based Permissions

Challenge: Fine-grained access control for admin features. **Solution**:

- Permission definitions stored in database
- Loaded via `PermissionsContext.js` on login
- Used throughout UI to conditionally render elements
- Checked in API routes via middleware

3. Booking System

Challenge: Managing seat availability and booking conflicts. **Solution**:

- Locking mechanism during seat selection
- Transaction-based booking process in backend
- Real-time seat status updates

4. Loyalty Program

Challenge: Calculating and managing loyalty points. **Solution**:

- Point accrual on successful bookings via observers
- Tier calculations in `LoyaltyService.php`
- Automatic membership creation for eligible users

Key Code Snippets

Authentication (Login.js)

```
const handleSubmit = async (e) => {
  e.preventDefault();
  try {
    const loginResponse = await loginUser(email, password);
    localStorage.setItem("token", loginResponse.access_token);
    setToken(loginResponse.access_token);
    const userData = await fetchUser();
    setUser(userData);

    const userPermissions = await loadPermissions(userData.role);
    if (hasAnyAdminPermissions(userPermissions)) {
      navigate("/admin", { replace: true });
    } else {
      navigate(redirectTo, { replace: true });
    }
  } catch (err) {
    console.error("Login error:", err);
    setErrorMessage("Login failed. Please try again.");
  }
};
```

Permission Check (Navbar.js)

```
const hasAnyAdminPermissions = useCallback(() => {
  if (!permissions || !user) return false;

  return Object.keys(permissions).length > 0;
}, [permissions, user]);

useEffect(() => {
  setNavAdmin(hasAnyAdminPermissions());
}, [user, permissions, hasAnyAdminPermissions]);
```

API Service (busService.js)

```
export const searchBuses = async (from, to, date) => {
  try {
    const response = await axios.get(`${API_URL}/buses/search`, {
      params: { from, to, date }
    });
    return response.data;
  } catch (error) {
    console.error("Error searching buses:", error);
    throw error;
  }
};
```

Database Model (BusTrip.php)

```
class BusTrip extends Model
{
    protected $fillable = [
        'bus_id', 'route_id', 'departure_date', 'departure_time',
        'arrival_time', 'price', 'status', 'available_seats'
    ];

    public function bus()
    {
        return $this->belongsTo(BusRegister::class, 'bus_id');
    }

    public function route()
    {
        return $this->belongsTo(BusRoute::class, 'route_id');
    }

    public function bookings()
    {
        return $this->hasMany(Booking::class, 'bus_id');
    }
}
```

Conclusion

The RS-Express Bus Booking System is a comprehensive web application built with Laravel and React, featuring robust authentication, role-based access control, and complex business logic for managing bus bookings. The architecture follows modern best practices with clear separation of concerns, modular components, and reusable services.

Key strengths of the system include:

- Modular architecture for maintainability
- Comprehensive role and permission system
- Responsive UI built with Tailwind CSS
- Reliable authentication with Laravel Sanctum
- Detailed booking and reporting capabilities
- Loyalty program for customer retention

Areas for potential enhancement include:

- Real-time notifications for booking status
- Mobile application development
- Advanced analytics and reporting
- Integration with additional payment gateways
- Performance optimization for scaling