Event Management Frontend System Report

EventSync Team

September 4, 2025

Contents

1 Executive Summary

This report details the Event Management Frontend, a React-based application designed to provide an intuitive interface for managing events, booking tickets, viewing notifications, and accessing analytics. Built with Vite, Tailwind CSS, and React Router, it integrates with a backend API to support user authentication, event browsing, ticket purchasing, and admin analytics. The frontend supports two user roles: users (for event browsing and ticket booking) and admins (for event and user management). This report outlines the system's architecture, features, components, setup instructions, and future enhancements, serving as a comprehensive guide for developers and stakeholders.

2 Project Overview

The Event Management Frontend is a modern web application that enables users to browse events, book tickets, and manage their profiles, while providing admins with tools to create events, analyze attendee data, and manage users. The application is responsive, visually appealing, and secure, with role-based access control and interactive visualizations for analytics.

2.1 Objectives

- Deliver a user-friendly interface for event browsing and ticket booking.
- Provide admins with robust tools for event management and analytics.
- Ensure secure authentication and role-based access.
- Offer interactive visualizations for attendee demographics and sales data.
- Support responsive design for desktop and mobile devices.

2.2 Target Audience

- **Users**: Individuals browsing events, booking tickets, and managing profiles.
- **Admins**: Event organizers managing events, users, and analytics.
- **Developers**: Teams integrating the frontend with the backend API.

3 Technical Architecture

The frontend is built using modern JavaScript technologies and follows a component-based architecture for modularity and reusability.

3.1 Technology Stack

- **React**: JavaScript library for building user interfaces.

- **Vite**: Fast build tool and development server.
- **Tailwind CSS**: Utility-first CSS framework for responsive styling.
- **React Router**: For client-side routing.
- **Axios**: For HTTP requests to the backend API.
- **JWT Decode**: For decoding JWT tokens to extract user roles.
- **QRCode**: For generating ticket QR codes.
- **Nivo**: For bar and lollipop chart visualizations.
- **Recharts**: For line and donut chart visualizations.
- **Lucide React**: Icon library for UI components.
- **React Icons**: Additional icons for event cards.

3.2 System Architecture

The application uses a component-based architecture with React:

- **Components**: Reusable UI elements (e.g., cards, charts, forms).
- **Pages**: Route-specific components (e.g., Login, AdminDashboard).
- **Services**: API interaction logic (e.g., authService, eventService).
- **Utilities**: Helper functions for authentication and QR code generation.
- **Routing**: React Router for navigation with private routes for authentication.

3.3 File Structure

```
src/├──
 assets/                    % Static assets (images, icons)├──
 common/                    % Reusable components (e.g., Button)├──
 pages/                     % Page components│├──
    AdminDashboard.jsx      % Admin dashboard│├──
    Analytics.jsx           % Analytics visualizations│├──
    Bookings.jsx            % Ticket booking page│├──
    EventCategories.jsx     % Event category management│├──
    EventDetailsPage.jsx    % Event details page│├──
    Login.jsx               % Login page│├──
    MyTickets.jsx           % User's tickets page│├──
    Notifications.jsx       % Notifications page│├──
    Register.jsx            % Registration page│├──
    Settings.jsx            % User settings page│├──
    Support.jsx             % Support page│├──
    UserEvent.jsx           % Event browsing page│├──
    AllAttendeeInsights.jsx % All events insights│├──
    SingleEventAttendeeInsights.jsx % Single event insights│└──
```

```
    AddEvent.jsx             % Add event page├──
services/│├──
    authService.js           % Authentication API calls│├──
    eventService.js          % Event API calls│└──
    notificationService.js   % Notification API calls├──
utils/│├──
    authUtils.js             % Authentication utilities│└──
    qrUtils.js               % QR code generation├──
components/│├──
    AnalyticsCards.jsx       % Analytics metric cards│├──
    AttendeeLocationsCard.jsx % Attendee locations table│├──
    BarChart.jsx             % Bar chart for locations│├──
    Card.jsx                 % Generic card│├──
    DonutChart.jsx           % Donut chart for analytics│├──
    EventCard.jsx            % Event display card│├──
    EventDetails.jsx         % Event details component│├──
    EventStatusLegend.jsx    % Event status legend│├──
    LatestEvent.jsx          % Latest event with seat map│├──
    LineChart.jsx            % Line chart for sales│├──
    LolipopChart.jsx         % Lollipop chart for attendee age│├──
    Navbar.jsx               % Event management header│├──
    Notifications.jsx        % Notifications component│├──
    RegisterForm.jsx         % Registration form│├──
    SeatPicker.jsx           % Seat selection component│└──
    SocialMediaCard.jsx      % Social media engagement├──
App.jsx                       % Main app with routing├──
main.jsx                      % React entry point└──
index.css                     % Global styles with Tailwind
```

4 Features

- **Authentication**: Login and registration with JWT-based access.
- **Event Management**: Browse, filter, search, and view event details.
- **Ticket Booking**: Interactive seat picker with QR code generation.
- **Notifications**: Display latest notifications with pagination support.
- **Analytics**: Visualize attendee demographics, sales, and social media engagement.
- **Seat Maps**: Display paid, reserved, and available seats.
- **Admin Tools**: Create/delete events, manage users, and view insights.
- **Responsive UI**: Optimized for desktop and mobile using Tailwind CSS.

5 Component Details

5.1 Page Components

- **AdminDashboard.jsx**: Admin dashboard with analytics and event management.
- **Analytics.jsx**: Displays charts for attendee and sales data.
- **Bookings.jsx**: Manages ticket bookings.
- **EventDetailsPage.jsx**: Shows detailed event information.
- **Login.jsx** and **Register.jsx**: Authentication pages.
- **MyTickets.jsx**: Displays user's booked tickets.
- **Notifications.jsx**: Shows user notifications.
- **AddEvent.jsx**: Form for creating new events (admin).
- **AllAttendeeInsights.jsx** and **SingleEventAttendeeInsights.jsx**: Analytics for all or specific events.
- **Settings.jsx**, **Support.jsx**, **Marketing.jsx**, **EventCategories.jsx**, **Users.jsx**: Admin and user management pages.

5.2 Reusable Components

- **AnalyticsCards.jsx**: Cards for metrics (age, gender, location, interests).
- **AttendeeLocationsCard.jsx**: Table for attendee locations.
- **BarChart.jsx**: Bar chart for location data.
- **Card.jsx**: Generic card for metrics.
- **DonutChart.jsx**: Donut chart for percentage-based analytics.
- **EventCard.jsx**: Displays event details with delete option (admin).
- **EventDetails.jsx**: Detailed event view with editable fields (admin).
- **EventStatusLegend.jsx**: Legend for event statuses (upcoming, pending, closed).
- **LatestEvent.jsx**: Latest event with seat map visualization.
- **LineChart.jsx**: Line chart for sales data with metrics.
- **LolipopChart.jsx**: Lollipop chart for attendee age distribution.
- **Navbar.jsx**: Header with filters, search, and admin controls.
- **Notifications.jsx**: Displays latest 5 notifications.
- **RegisterForm.jsx**: Form for user registration.
- **SeatPicker.jsx**: Interactive seat selection grid.

- **SocialMediaCard.jsx**: Social media engagement metrics.

5.3 Utilities and Services

- **authService.js**: Handles login and register API calls.
- **api.js**: Configures Axios with JWT token interceptors.
- **authUtils.js**: Manages authentication and role checks.
- **qrUtils.js**: Generates QR codes for tickets.

6 Setup Instructions

6.1 Prerequisites

- Node.js (v14 or higher)
- npm or yarn
- Backend API running at `http://localhost:5000/api`

6.2 Installation

1. Clone the repository:

```
git clone <repository-url>
cd event-management-frontend
```

2. Install dependencies:

```
npm install
```

3. Create a `.env` file:

```
VITE_API_URL=http://localhost:5000/api
```

4. Start the development server:

```
npm run dev
```

7 API Integration

The frontend communicates with the backend via Axios, with key endpoints:

- **Authentication**: `/auth/login`, `/auth/register`.
- **Events**: `/events`, `/events/:id`, `/events/tickets`.
- **Notifications**: `/notifications`.

Tokens are stored in `localStorage` and added to requests via Axios interceptors.

8 Security

- **Authentication**: JWT tokens stored in `localStorage`.
- **Role-Based Access**: `PrivateRoute` restricts routes by role.
- **Input Validation**: Handled by backend, with frontend displaying errors.

9 Future Improvements

- Add unit and integration tests with Jest/React Testing Library.
- Implement real-time notifications using WebSockets.
- Enhance accessibility with ARIA labels and keyboard navigation.
- Add loading spinners and skeleton screens.
- Support internationalization (i18n).
- Optimize performance with lazy loading.

10 Conclusion

The Event Management Frontend provides a modern, responsive interface for event and ticket management, with robust features for users and admins. Its modular architecture and integration with the backend API ensure scalability and ease of maintenance. Future enhancements will further improve its functionality and user experience.