Fiesta Venue Management System

Complete Technical Documentation

Version: 2.0
Last Updated: November 2024
Status: Production Ready

Table of Contents

1. Introduction
2. Getting Started
3. Architecture Overview
4. Design System
5. Project Structure
6. Core Features
7. API Integration
8. Component Library
9. State Management
10. Routing & Navigation
11. Authentication & Authorization
12. Data Fetching Patterns
13. Form Handling
14. Error Handling
15. Performance Optimization
16. Testing Strategy
17. Deployment
18. Troubleshooting
19. Best Practices
20. API Reference

Introduction

What is Fiesta?

Fiesta is a comprehensive venue management system designed for event venues, wedding halls, conference centers, and hospitality businesses. It provides a complete solution for managing events, clients, partners, payments, and team operations.

Key Features

- 📅 **Event Management** - Full event lifecycle tracking
- 👥 **Client Management** - Client database with history
- 🤝 **Partner Management** - Vendor and supplier tracking
- 💰 **Payment Tracking** - Invoice and payment management
- 📊 **Financial Analytics** - Revenue, expenses, and P&L reports
- ✅ **Task Management** - Team task assignment and tracking
- 🔔 **Reminders** - Automated event and payment reminders
- 👨‍💼 **Team Collaboration** - Role-based access control
- 📱 **Responsive Design** - Works on desktop, tablet, and mobile
- 🌓 **Dark Mode** - Eye-friendly dark theme support

Technology Stack

Frontend:

- React 18.2+ with Hooks
- Vite 5.x (Build tool)
- React Router DOM v6 (Routing)
- Tailwind CSS v3 (Styling)
- Lucide React (Icons)
- Axios (HTTP Client)
- React Calendar (Calendar view)
- React Hot Toast (Notifications)

Development Tools:

- ESLint (Code linting)
- Prettier (Code formatting)
- React DevTools (Debugging)

Getting Started

Prerequisites

- Node.js 18.x or higher
- npm 9.x or higher
- Git
- Code editor (VS Code recommended)

Installation

bash

```bash
# Clone the repository
git clone https://github.com/your-org/fiesta-frontend.git
cd fiesta-frontend

# Install dependencies
npm install

# Copy environment file
cp .env.example .env

# Configure environment variables
nano .env
```

Environment Variables

Create a `.env` file in the root directory:

env

```env
# API Configuration
VITE_API_URL=http://localhost:5000/api/v1
VITE_API_TIMEOUT=30000

# Application
VITE_APP_NAME=Fiesta
VITE_APP_VERSION=2.0.0

# Features
VITE_ENABLE_ANALYTICS=false
VITE_ENABLE_DEBUG=true

# External Services (if applicable)
VITE_STRIPE_PUBLIC_KEY=pk_test_xxx
VITE_GOOGLE_MAPS_API_KEY=AIzaxxx
```

Running the Application

bash

```bash
# Development mode (with hot reload)
npm run dev

# Production build
npm run build

# Preview production build
npm run preview

# Lint code
npm run lint

# Format code
npm run format
```

The application will be available at:

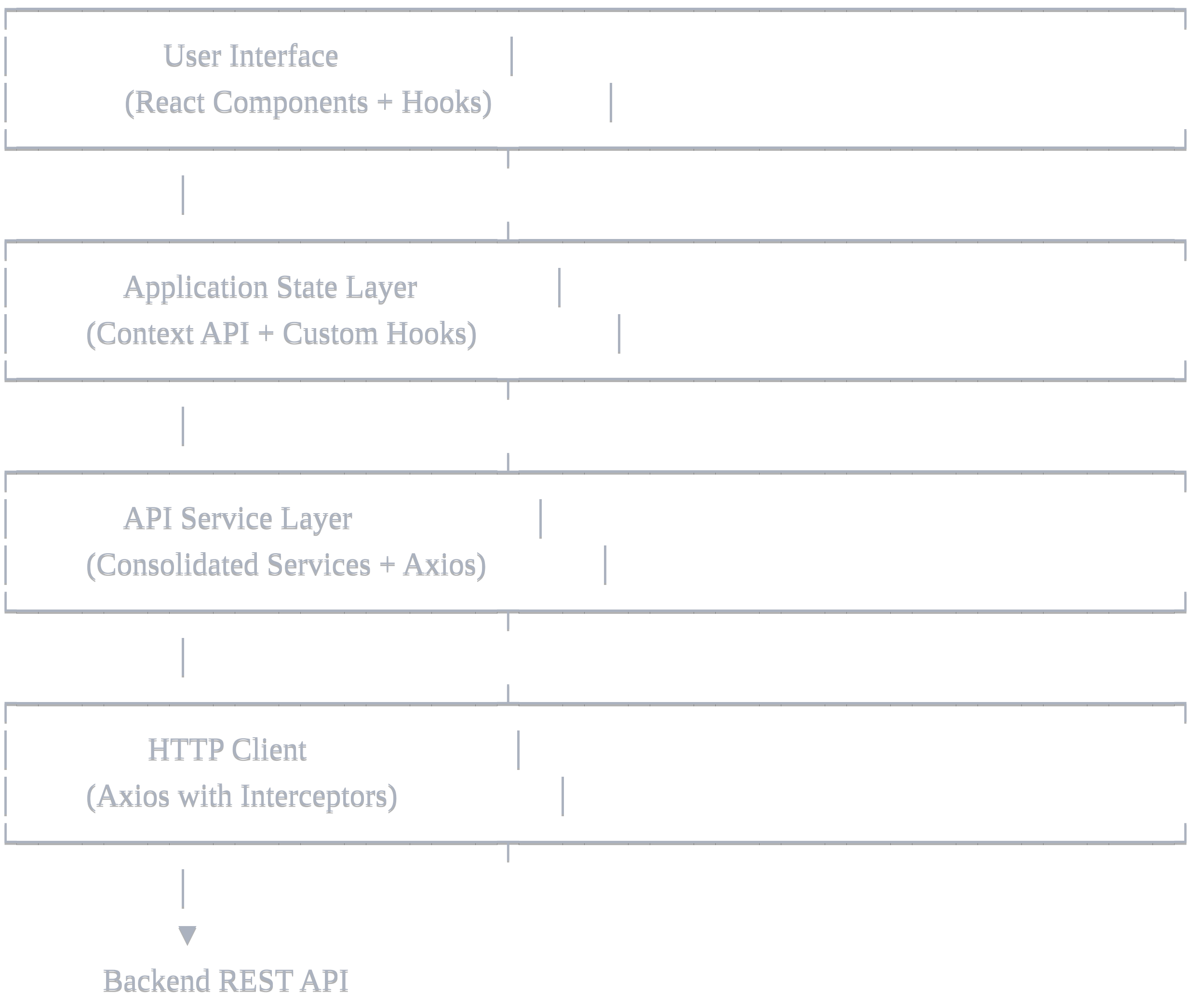
- Development: `http://localhost:5173`
- Preview: `http://localhost:4173`

First Time Setup

1. **Start the backend API** (see backend documentation)
2. **Register a new venue account**
3. **Complete venue profile setup**
4. **Invite team members**
5. **Configure venue settings**

Architecture Overview

High-Level Architecture


User Interface
(React Components + Hooks)
Application State Layer
(Context API + Custom Hooks)
API Service Layer
(Consolidated Services + Axios)
HTTP Client
(Axios with Interceptors)
Backend REST API

Design Principles

1. **Separation of Concerns** - Each layer has a single responsibility
2. **Component Composition** - Small, reusable components
3. **Declarative UI** - React's declarative approach
4. **Unidirectional Data Flow** - Props down, events up
5. **Mobile-First** - Responsive design from the ground up
6. **Accessibility** - WCAG 2.1 AA compliance
7. **Performance** - Code splitting and lazy loading

Application Flow

```
1. User Authentication
   ├── Login/Register
   ├── Token Storage
   └── Route Protection

2. Main Application
   ├── Layout Rendering
   │   ├── Sidebar
   │   ├── TopBar
   │   └── Main Content
   │
   ├── Data Fetching
   │   ├── API Call
   │   ├── Loading State
   │   └── Data Display
   │
   └── User Interactions
       ├── Form Submission
       ├── CRUD Operations
       └── Navigation
```

Design System

Brand Identity

Primary Brand Color:

- Orange 500: `#F97316`
- Used for: Buttons, links, active states, brand elements

Color Palette:

css

```css
/* Primary Colors */
--orange-50: #FFF7ED;
--orange-500: #F97316;
--orange-600: #EA580C;
--orange-700: #C2410C;

/* Semantic Colors */
--green-50: #ECFDF5;
--green-500: #10B981;   /* Success */
--yellow-50: #FFFBEB;
--yellow-500: #F59E0B;  /* Warning */
--red-50: #FEF2F2;
--red-500: #EF4444;     /* Error */
--blue-50: #EFF6FF;
--blue-500: #3B82F6;    /* Info */

/* Neutral Colors */
--gray-50: #F9FAFB;     /* Page background */
--gray-100: #F3F4F6;    /* Card hover */
--gray-200: #E5E7EB;    /* Borders */
--gray-300: #D1D5DB;    /* Dividers */
--gray-400: #9CA3AF;    /* Disabled */
--gray-500: #6B7280;    /* Placeholder */
--gray-600: #4B5563;    /* Secondary text */
--gray-700: #374151;    /* Body text */
--gray-900: #111827;    /* Headings */
--white: #FFFFFF;       /* Cards, panels */
```

Dark Mode Colors

css

```css
/* Dark Mode Overrides */
--dark-bg-primary: #0F172A;     /* Main background */
--dark-bg-secondary: #1E293B;   /* Cards */
--dark-bg-tertiary: #334155;    /* Hover states */
--dark-text-primary: #F1F5F9;   /* Headings */
--dark-text-secondary: #CBD5E1; /* Body text */
--dark-text-tertiary: #94A3B8;  /* Muted text */
--dark-border: #334155;         /* Borders */
```

Typography

Font Family:

css

```css
font-family: ui-sans-serif, system-ui, -apple-system,
             BlinkMacSystemFont, "Segoe UI", Roboto,
             "Helvetica Neue", Arial, sans-serif;
```

Font Sizes:

Name	Size	Tailwind	Usage
Display	30px	text-3xl	Page titles
H1	24px	text-2xl	Section titles
H2	20px	text-xl	Card titles
H3	18px	text-lg	Subsections
Body Large	16px	text-base	Primary text
Body	14px	text-sm	Secondary text
Caption	12px	text-xs	Labels, meta

Font Weights:

- Bold: `700` - Headings, important numbers
- Semibold: `600` - Subheadings, emphasis
- Medium: `500` - Labels, buttons
- Regular: `400` - Body text

Spacing System

Based on 4px grid (Tailwind scale):

css

```css
/* Spacing Scale */
0.5 → 2px   (gap-0.5, p-0.5)
1   → 4px   (gap-1, p-1)
2   → 8px   (gap-2, p-2)
3   → 12px  (gap-3, p-3)
4   → 16px  (gap-4, p-4)
6   → 24px  (gap-6, p-6)
8   → 32px  (gap-8, p-8)
12  → 48px  (gap-12, p-12)
16  → 64px  (gap-16, p-16)
```

Common Spacing Patterns:

jsx

```jsx
// Card padding
<div className="p-6">       {/* 24px all sides */}

// Section spacing
<div className="space-y-6">  {/* 24px vertical gap */}

// List items
<div className="space-y-2">  {/* 8px vertical gap */}

// Form fields
<div className="space-y-4">  {/* 16px vertical gap */}

// Page container
<div className="p-4 sm:p-6 lg:p-8">  {/* Responsive */}
```

Border Radius

css

```css
rounded-sm   → 2px   (small elements)
rounded      → 4px   (buttons, inputs)
rounded-lg   → 8px   (cards, panels)
rounded-xl   → 12px  (large cards)
rounded-2xl  → 16px  (modals)
rounded-full → 9999px (badges, avatars)
```

Shadows

css

```
/* None (default for cards) */
border border-gray-200

/* Light (hover state) */
hover:shadow-md

/* Medium (dropdowns) */
shadow-lg

/* Heavy (modals) */
shadow-xl

/* Focus (inputs) */
focus:ring-2 focus:ring-orange-500
```

Icons

Library: Lucide React
Default Size: 20px (`h-5 w-5`)
Large Icons: 24px (`h-6 w-6`)
Empty States: 48px (`h-12 w-12`)

Common Icons:

jsx

```
import {
  Home,          // Dashboard
  Calendar,      // Events
  Users,         // Clients
  Briefcase,     // Partners
  CreditCard,    // Payments
  DollarSign,    // Finance
  CheckSquare,   // Tasks
  Bell,          // Reminders
  Settings,      // Settings
  Plus,          // Create
  Edit,          // Edit
  Trash2,        // Delete
  Search,        // Search
  Filter,        // Filter
  Download,      // Export
} from 'lucide-react';
```

Responsive Breakpoints

css

```css
/* Tailwind Defaults */
sm:  640px  /* Small tablets */
md:  768px  /* Tablets */
lg:  1024px /* Desktop */
xl:  1280px /* Large desktop */
2xl: 1536px /* Extra large */
```

Mobile-First Approach:

jsx

```
// Base styles = mobile
// Add sm:, md:, lg: for larger screens

<div className="
  text-sm      {/* Mobile: 14px */}
  md:text-base {/* Tablet+: 16px */}
  lg:text-lg   {/* Desktop+: 18px */}
">
```

Project Structure

Directory Organization

```
fiesta-frontend/
├── public/                 # Static assets
│   ├── favicon.ico
│   ├── logo.png
│   └── robots.txt
│
├── src/
│   ├── api/                # API layer
│   │   ├── services/
│   │   │   └── index.js      # Consolidated services
│   │   └── axios.js          # Axios configuration
│   │
│   ├── assets/             # Images, fonts, etc.
│   │   ├── images/
│   │   ├── icons/
│   │   └── fonts/
│   │
│   ├── components/          # Reusable components
│   │   ├── common/          # Shared UI components
│   │   │   ├── Button.jsx
│   │   │   ├── Input.jsx
│   │   │   ├── Modal.jsx
│   │   │   ├── Table.jsx
│   │   │   ├── Badge.jsx
│   │   │   ├── Card.jsx
│   │   │   ├── EmptyState.jsx
│   │   │   ├── LoadingSpinner.jsx
│   │   │   ├── Pagination.jsx
│   │   │   ├── SearchBar.jsx
│   │   │   ├── FilterDropdown.jsx
│   │   │   └── ConfirmDialog.jsx
│   │   │
│   │   ├── layout/          # Layout components
│   │   │   ├── Sidebar.jsx
│   │   │   ├── TopBar.jsx
│   │   │   ├── MainLayout.jsx
│   │   │   └── Footer.jsx
│   │   │
│   │   └── features/        # Feature-specific components
│   │       ├── events/
│   │       ├── clients/
│   │       ├── partners/
│   │       └── payments/
│   │
│   ├── context/           # React Context providers
│   │   ├── AuthContext.jsx
│   │   ├── ThemeContext.jsx
│   │   └── NotificationContext.jsx
│   │
│   ├── hooks/             # Custom React hooks
│   │   ├── useApi.js
│   │   ├── useAuth.js
│   │   ├── useTheme.js
│   │   ├── useDebounce.js
│   │   ├── useLocalStorage.js
│   │   └── usePermissions.js
│   │
│   ├── pages/             # Page components
│   │   ├── auth/
│   │   │   ├── Login.jsx
│   │   │   ├── Register.jsx
│   │   │   ├── ForgotPassword.jsx
│   │   │   └── ResetPassword.jsx
│   │   │
│   │   ├── Dashboard.jsx
│   │   │
│   │   ├── events/
│   │   │   ├── EventsList.jsx
│   │   │   ├── EventDetail.jsx
│   │   │   ├── EventForm.jsx
│   │   │   └── EventCalendar.jsx
│   │   │
│   │   ├── clients/
│   │   │   ├── ClientsList.jsx
│   │   │   ├── ClientDetail.jsx
│   │   │   └── ClientForm.jsx
│   │   │
│   │   ├── partners/
│   │   │   ├── PartnersList.jsx
│   │   │   ├── PartnerDetail.jsx
│   │   │   └── PartnerForm.jsx
│   │   │
│   │   ├── payments/
│   │   │   ├── PaymentsList.jsx
│   │   │   ├── PaymentDetail.jsx
│   │   │   └── PaymentForm.jsx
│   │   │
│   │   ├── finance/
│   │   │   ├── FinanceOverview.jsx
│   │   │   ├── FinanceReports.jsx
│   │   │   └── FinanceForm.jsx
│   │   │
│   │   ├── tasks/
│   │   │   ├── TasksList.jsx
│   │   │   ├── TaskBoard.jsx
│   │   │   └── TaskDetail.jsx
│   │   │
│   │   ├── reminders/
│   │   │   ├── RemindersList.jsx
│   │   │   └── ReminderForm.jsx
│   │   │
│   │   ├── team/
│   │   │   ├── TeamList.jsx
│   │   │   ├── TeamMemberDetail.jsx
│   │   │   └── InviteTeam.jsx
│   │   │
│   │   ├── roles/
│   │   │   ├── RolesList.jsx
│   │   │   └── RoleForm.jsx
│   │   │
│   │   └── settings/
│   │       ├── VenueSettings.jsx
│   │       ├── ProfileSettings.jsx
│   │       └── SecuritySettings.jsx
│   │
│   ├── routes/              # Routing configuration
│   │   ├── AppRoutes.jsx
│   │   ├── ProtectedRoute.jsx
│   │   └── PublicRoute.jsx
│   │
│   ├── utils/               # Utility functions
│   │   ├── formatters.js   # Date, currency, text formatters
│   │   ├── validators.js   # Form validation
│   │   ├── constants.js    # App constants
│   │   ├── helpers.js      # Helper functions
│   │   └── permissions.js  # Permission checks
│   │
│   ├── styles/              # Global styles
│   │   ├── index.css       # Global CSS
│   │   └── tailwind.css    # Tailwind imports
│   │
│   ├── App.jsx              # Root component
│   └── main.jsx          # Entry point
│
├── .env                  # Environment variables
├── .env.example          # Example env file
├── .eslintrc.json        # ESLint configuration
├── .prettierrc           # Prettier configuration
├── .gitignore            # Git ignore rules
├── index.html            # HTML template
├── package.json          # Dependencies
├── postcss.config.js     # PostCSS config
├── tailwind.config.js    # Tailwind config
├── vite.config.js        # Vite config
└── README.md             # Project README
```

File Naming Conventions

Components:

- PascalCase: `EventCard.jsx`, `UserProfile.jsx`
- Feature folders: lowercase `events/`, `clients/`

Utilities:

- camelCase: `formatDate.js`, `validateEmail.js`

Styles:

- kebab-case: `custom-styles.css`

Constants:

- UPPER_SNAKE_CASE in files: `API_ENDPOINTS`, `USER_ROLES`

Core Features

1. Dashboard

Purpose: Central hub showing key metrics and recent activity

Key Metrics:

- Total Events (with monthly change %)
- Revenue (with monthly change %)
- Active Clients (with monthly change %)
- Pending Payments (with monthly change %)

Sections:

1. **Stats Cards** - 4-column grid (responsive to 1 column on mobile)
2. **Upcoming Events** - Next 5 events with status badges
3. **Recent Payments** - Latest 5 payment activities

Data Sources:

javascript

```javascript
// Dashboard loads data from:
dashboardService.getStats()           // Stats cards
dashboardService.getUpcomingEvents()    // Events section
dashboardService.getRecentPayments()    // Payments section
```

Features:

- Real-time data refresh
- Loading skeletons
- Empty states
- Auto-refresh every 5 minutes
- Manual refresh button

2. Event Management

Purpose: Complete event lifecycle management

Event States:

- **Draft** - Initial creation
- **Pending** - Awaiting confirmation
- **Confirmed** - Locked and scheduled
- **In Progress** - Currently happening
- **Completed** - Successfully finished
- **Cancelled** - Cancelled by client/venue

Key Features:

- Full CRUD operations
- Calendar view (monthly/weekly/daily)
- Event templates
- Status workflow
- Client assignment
- Partner assignment (catering, DJ, etc.)
- Payment schedule
- Task creation from events
- Document attachments
- Event history log

Views:

1. **List View** - Table with filtering and sorting
2. **Calendar View** - Visual month/week/day view
3. **Detail View** - Single event with full info

Filtering:

- By status
- By date range
- By client
- By venue space

- By event type

3. Client Management

Purpose: Maintain client database and relationship history

Client Information:

- Basic info (name, email, phone, address)
- Company details (optional)
- Client type (individual/corporate)
- Contact preferences
- Tags/categories
- Notes and history

Features:

- Full CRUD operations
- Client portal (future)
- Event history per client
- Payment history per client
- Client statistics
- Import/export clients
- Email/SMS integration (future)
- Client ratings and feedback

Analytics:

- Total revenue per client
- Number of events per client
- Average event value
- Client lifetime value
- Last event date

4. Partner Management

Purpose: Manage vendors and service providers

Partner Types:

- Catering
- Photography/Videography
- DJ/Entertainment
- Decorations
- Transportation
- Security
- Cleaning
- Equipment Rental

Partner Information:

- Company name
- Contact person
- Services offered
- Pricing structure
- Availability calendar
- Performance ratings
- Contract details
- Payment terms

Features:

- Full CRUD operations
- Partner portal (future)
- Service catalog
- Availability checking
- Performance tracking
- Partner comparison
- Contract management
- Payment tracking to partners

5. Payment & Invoicing

Purpose: Track all financial transactions

Payment Types:

- Event deposit
- Event balance
- Additional services
- Refunds
- Partner payments

Payment Methods:

- Cash
- Bank transfer
- Credit card
- Check
- Online payment

Payment States:

- Pending
- Paid
- Partially paid
- Overdue
- Refunded
- Cancelled

Features:

- Invoice generation
- Payment reminders
- Receipt generation
- Refund processing
- Payment plans
- Auto payment matching
- Payment history
- Export transactions

6. Financial Management

Purpose: Track income, expenses, and profitability

Transaction Categories:

Income:

- Event revenue

- Additional services
- Late fees
- Other income

Expenses:

- Partner payments
- Utilities
- Salaries
- Maintenance
- Marketing
- Supplies
- Other expenses

Reports:

1. **Cash Flow** - Money in vs money out
2. **P&L Statement** - Profit and loss
3. **Revenue Trends** - Monthly/quarterly trends
4. **Expense Breakdown** - Category analysis
5. **Tax Summary** - For tax filing

Features:

- Transaction logging
- Receipt uploads
- Budget tracking
- Financial forecasting
- Export to Excel/CSV
- Multi-currency support (future)
- Integration with accounting software (future)

7. Task Management

Purpose: Team task assignment and tracking

Task Properties:

- Title and description
- Assigned to (team member)
- Due date
- Priority (low/medium/high)
- Status (to-do/in-progress/done)
- Related event (optional)
- Subtasks
- Comments
- Attachments

Views:

1. **List View** - Filterable task list
2. **Board View** - Kanban board (To-Do, In Progress, Done)
3. **My Tasks** - Current user's tasks only
4. **Calendar View** - Tasks by due date

Features:

- Task templates
- Recurring tasks
- Task dependencies

- Time tracking (future)
- Task comments/chat
- File attachments
- Email notifications
- Task activity log

8. Reminder System

Purpose: Automated notifications for important events

Reminder Types:

- Event approaching
- Payment due
- Task deadline
- Follow-up reminder
- Custom reminder

Reminder Triggers:

- Time-based (X days before)
- Event-based (status change)
- Manual (user-created)

Delivery Methods:

- In-app notification
- Email
- SMS (future)
- Push notification (future)

Features:

- Snooze reminders
- Mark as complete
- Recurring reminders
- Reminder templates
- Reminder history
- Batch reminders

9. Team Management

Purpose: Manage team members and permissions

User Roles:

- **Owner** - Full access
- **Admin** - All features except billing
- **Manager** - Events, clients, partners
- **Staff** - Limited access
- **Custom** - Define specific permissions

Permissions:

- View/Create/Edit/Delete per module
- Financial access (yes/no)
- Export data (yes/no)
- Invite team (yes/no)
- Settings access (yes/no)

Features:

- Team invitations (email-based)
- Role assignment
- Permission management
- Activity logs
- Performance metrics
- Availability calendar
- Team chat (future)

10. Settings

Venue Settings:

- Venue name and logo
- Contact information
- Operating hours
- Venue capacity
- Venue spaces/rooms
- Amenities
- Pricing structure

Profile Settings:

- Personal information
- Password change
- Email preferences
- Notification settings
- Two-factor authentication

System Settings:

- Time zone
- Date format
- Currency
- Language (future)
- Integrations
- API access

API Integration

API Service Architecture

Location: `src/api/services/index.js`

All API calls go through centralized service layer:

javascript

```
import { eventService, clientService } from '@/api/services';

// All services return consistent structure
const { events, pagination } = await eventService.getAll();
```

Axios Configuration

Location: `src/api/axios.js`

Features:

- Base URL configuration
- Request timeout (30s)
- Auth token injection
- Venue ID header for multi-tenancy
- Request/response logging (dev mode)
- Error normalization
- Auto-logout on 401

Configuration:

javascript

```javascript
const api = axios.create({
  baseURL: import.meta.env.VITE_API_URL,
  timeout: 30000,
  headers: {
    'Content-Type': 'application/json',
  },
});
```

Request Interceptor

javascript

```
api.interceptors.request.use((config) => {
  // Add auth token
  const token = localStorage.getItem('token');
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }

  // Add venue ID for multi-tenancy
  const venueId = localStorage.getItem('venueId');
  if (venueId) {
    config.headers['X-Venue-ID'] = venueId;
  }

  return config;
});
```

Response Interceptor

javascript

```javascript
api.interceptors.response.use(
  (response) => response,
  (error) => {
    if (error.response?.status === 401) {
      // Auto-logout and redirect
      localStorage.clear();
      window.location.href = '/login';
    }

    // Return normalized error
    return Promise.reject({
      status: error.response?.status,
      message: error.response?.data?.message,
      errors: error.response?.data?.errors,
    });
  }
);
```

Service Methods

Each service follows this pattern:

javascript

```javascript
export const eventService = {
  // List with filters
  getAll: async (params = {}) => {
    const response = await api.get('/events', { params });
    return handleResponse(response);
  },

  // Single item
  getById: async (id) => {
    const response = await api.get(`/events/${id}`);
    return handleResponse(response);
  },

  // Create
  create: async (data) => {
    const response = await api.post('/events', data);
    return handleResponse(response);
  },

  // Update
  update: async (id, data) => {
    const response = await api.put(`/events/${id}`, data);
    return handleResponse(response);
  },

  // Delete
  delete: async (id) => {
    const response = await api.delete(`/events/${id}`);
    return handleResponse(response);
  },
};
```

Response Structure

All API responses follow this structure:

javascript

```
{
  success: true,
  message: "Operation successful",
  data: {
    // Actual data here
    events: [...],
    pagination: {
      page: 1,
      limit: 10,
      total: 100,
      pages: 10
    }
  }
}
```

The service layer extracts `data` for you:

javascript

```javascript
// You get this directly:
const { events, pagination } = await eventService.getAll();
```

Error Structure

Errors are normalized:

javascript

```javascript
{
  status: 422,
  message: "Validation failed",
  errors: {
    email: "Email is required",
    password: "Password must be at least 8 characters"
  }
}
```

Component Library

Common Components

Button

Location: `src/components/common/Button.jsx`

Variants:

- `primary` - Orange background (default)
- `secondary` - Gray background
- `outline` - Transparent with border
- `ghost` - No background
- `danger` - Red (for delete actions)

Sizes:

- `sm` - Small (32px height)
- `md` - Medium (40px height, default)
- `lg` - Large (48px height)

Usage:

jsx

```jsx
import Button from '@/components/common/Button';

<Button
  variant="primary"
  size="md"
  onClick={handleClick}
  loading={isLoading}
  disabled={isDisabled}
  icon={<PlusIcon />}
>
  Create Event
</Button>
```

Input

Location: `src/components/common/Input.jsx`

Types:

- `text` - Text input
- `email` - Email input
- `password` - Password input
- `number` - Number input
- `date` - Date picker
- `textarea` - Multi-line text

Features:

- Label support
- Error message display
- Helper text
- Left/right icons
- Disabled state
- Required indicator

Usage:

jsx

```jsx
import Input from '@/components/common/Input';

<Input
  label="Event Name"
  type="text"
  value={name}
  onChange={(e) => setName(e.target.value)}
  error={errors.name}
  placeholder="Enter event name"
  required
  icon={<CalendarIcon />}
/>
```

Modal

Location: `src/components/common/Modal.jsx`

Features:

- Backdrop overlay
- Close on escape key
- Close on backdrop click
- Header with title
- Footer with actions
- Scrollable content
- Multiple sizes

Usage:

jsx

```jsx
import Modal from '@/components/common/Modal';

<Modal
  isOpen={isOpen}
  onClose={handleClose}
  title="Create New Event"
  size="lg"
>
  <div className="p-6">
    {/* Modal content */}
  </div>

  <div className="flex gap-2 p-6 border-t">
    <Button onClick={handleClose} variant="outline">
      Cancel
    </Button>
    <Button onClick={handleSubmit} loading={loading}>
      Create Event
    </Button>
  </div>
</Modal>
```

Table

Location: `src/components/common/Table.jsx`

Features:

- Sortable columns
- Row selection
- Pagination
- Loading state
- Empty state
- Responsive (stacks on mobile)
- Row actions

Usage:

jsx

```jsx
import Table from '@/components/common/Table';

const columns = [
  { key: 'name', label: 'Event Name', sortable: true },
  { key: 'date', label: 'Date', sortable: true },
  { key: 'status', label: 'Status', render: (row) => <Badge>{row.status}</Badge> },
  { key: 'actions', label: 'Actions', render: (row) => <ActionsMenu /> },
];

<Table
  columns={columns}
  data={events}
  loading={loading}
  onSort={handleSort}
  onRowClick={handleRowClick}
  emptyMessage="No events found"
/>
```

Badge

Location: `src/components/common/Badge.jsx`

Variants:

- `success` - Green (confirmed, paid)
- `warning` - Yellow (pending, in-progress)
- `danger` - Red (cancelled, overdue)
- `info` - Blue (completed)
- `default` - Gray (draft, inactive)

Usage:

jsx

```jsx
import Badge from '@/components/common/Badge';

<Badge variant="success">Confirmed</Badge>
<Badge variant="warning">Pending</Badge>
<Badge variant="danger">Cancelled</Badge>
```

Card

Location: `src/components/common/Card.jsx`

Features:

- Header with title
- Optional subtitle
- Action button in header
- Footer section
- Hover effect
- Clickable variant

Usage:

jsx

```jsx
import Card from '@/components/common/Card';

<Card
  title="Upcoming Events"
  subtitle="Next 7 days"
  action={<Button size="sm">View All</Button>}
  footer={<div className="text-sm text-gray-500">Updated 5 min ago</div>}
>
  {/* Card content */}
</Card>
```

EmptyState

Location: `src/components/common/EmptyState.jsx`

Usage:

jsx

```jsx
import EmptyState from '@/components/common/EmptyState';
import { CalendarIcon } from 'lucide-react';

<EmptyState
  icon={<CalendarIcon />}
  title="No events yet"
  description="Create your first event to get started"
  action={
    <Button onClick={handleCreate}>
      Create Event
    </Button>
  }
/>
```

LoadingSpinner

Location: `src/components/common/LoadingSpinner.jsx`

Sizes:

- `sm` - 16px
- `md` - 24px (default)
- `lg` - 32px
- `xl` - 48px

Usage:

jsx

```jsx
import LoadingSpinner from '@/components/common/LoadingSpinner';

<LoadingSpinner size="lg" />

// Or full-page loader
<div className="flex items-center justify-center min-h-screen">
  <LoadingSpinner size="xl" />
</div>
```

Pagination

Location: `src/components/common/Pagination.jsx`

Features:

- Page numbers
- Previous/Next buttons
- Page size selector
- Jump to page
- Results count

Usage:

jsx

```jsx
import Pagination from '@/components/common/Pagination';

<Pagination
  currentPage={page}
  totalPages={pagination.pages}
  totalItems={pagination.total}
  pageSize={pagination.limit}
  onPageChange={handlePageChange}
  onPageSizeChange={handlePageSizeChange}
/>
```

SearchBar

Location: `src/components/common/SearchBar.jsx`

Features:

- Debounced search
- Clear button
- Loading indicator
- Keyboard shortcuts (Cmd+K)

Usage:

jsx

```jsx
import SearchBar from '@/components/common/SearchBar';

<SearchBar
  value={searchQuery}
  onChange={handleSearch}
  placeholder="Search events..."
  loading={searching}
/>
```

ConfirmDialog

Location: `src/components/common/ConfirmDialog.jsx`

Usage:

jsx

```
import ConfirmDialog from '@/components/common/ConfirmDialog';

<ConfirmDialog
  isOpen={showConfirm}
  onClose={() => setShowConfirm(false)}
  onConfirm={handleDelete}
  title="Delete Event"
  message="Are you sure you want to delete this event? This action cannot be undone."
  confirmText="Delete"
  confirmVariant="danger"
  loading={deleting}
/>
```

State Management

Context API

We use React Context for global state management:

AuthContext

Location: `src/context/AuthContext.jsx`

State:

- `user` - Current user object
- `loading` - Auth loading state
- `isAuthenticated` - Boolean auth status

Methods:

- `login(email, password)` - Authenticate user
- `logout()` - Clear session
- `updateUser(data)` - Update user info

Usage:

jsx

```jsx
import { useAuth } from '@/context/AuthContext';

function MyComponent() {
  const { user, isAuthenticated, login, logout } = useAuth();

  if (!isAuthenticated) {
    return <LoginPrompt />;
  }

  return <div>Welcome, {user.name}!</div>;
}
```

ThemeContext

Location: `src/context/ThemeContext.jsx`

State:

- `theme` - 'light' or 'dark'
- `toggleTheme()` - Switch themes

Usage:

jsx

```jsx
import { useTheme } from '@/context/ThemeContext';

function ThemeToggle() {
  const { theme, toggleTheme } = useTheme();

  return (
    <button onClick={toggleTheme}>
      {theme === 'light' ? <MoonIcon /> : <SunIcon />}
    </button>
  );
}
```

NotificationContext

Location: `src/context/NotificationContext.jsx`

Methods:

- `showNotification(message, type)` - Show toast
- `hideNotification(id)` - Hide specific toast
- `clearAll()` - Clear all toasts

Usage:

jsx

```jsx
import { useNotification } from '@/context/NotificationContext';

function MyComponent() {
  const { showNotification } = useNotification();

  const handleSuccess = () => {
    showNotification('Event created successfully!', 'success');
  };

  const handleError = () => {
    showNotification('Failed to create event', 'error');
  };

  return <Button onClick={handleSuccess}>Create</Button>;
}
```

Custom Hooks

useApi

Location: `src/hooks/useApi.js`

Automatic data fetching with loading/error states:

jsx

```jsx
import { useApi } from '@/hooks/useApi';
import { eventService } from '@/api/services';

function EventsList() {
  const { data, loading, error, refetch } = useApi(
    () => eventService.getAll({ status: 'active' })
  );

  if (loading) return <LoadingSpinner />;
  if (error) return <ErrorMessage error={error} />;

  return (
    <div>
      <Button onClick={refetch}>Refresh</Button>
      {data.events.map(event => (
        <EventCard key={event.id} event={event} />
      ))}
    </div>
  );
}
```

useApiMutation

Location: `src/hooks/useApi.js`

For create/update/delete operations:

jsx

```
import { useApiMutation } from '@/hooks/useApi';
import { eventService } from '@/api/services';
import { toast } from 'react-hot-toast';

function CreateEventForm() {
  const navigate = useNavigate();

  const { mutate: createEvent, loading } = useApiMutation(
    eventService.create,
    {
      onSuccess: (result) => {
        toast.success('Event created!');
        navigate(`/events/${result.event.id}`);
      },
      onError: (error) => {
        toast.error(error.message);
      }
    }
  );

  const handleSubmit = async (formData) => {
    await createEvent(formData);
  };

  return (
    <form onSubmit={handleSubmit}>
      {/* form fields */}
      <Button type="submit" loading={loading}>
        Create Event
      </Button>
    </form>
  );
}
```

useDebounce

Location: `src/hooks/useDebounce.js`

Debounce search inputs:

jsx

```
import { useDebounce } from '@/hooks/useDebounce';
import { useState, useEffect } from 'react';

function SearchEvents() {
  const [query, setQuery] = useState('');
  const debouncedQuery = useDebounce(query, 500); // 500ms delay

  useEffect(() => {
    if (debouncedQuery) {
      // Perform search
      searchEvents(debouncedQuery);
    }
  }, [debouncedQuery]);

  return (
    <input
      value={query}
      onChange={(e) => setQuery(e.target.value)}
      placeholder="Search..."
    />
  );
}
```

useLocalStorage

Location: `src/hooks/useLocalStorage.js`

Persist state to localStorage:

jsx

```
import { useLocalStorage } from '@/hooks/useLocalStorage';

function MyComponent() {
  const [filters, setFilters] = useLocalStorage('eventFilters', {
    status: 'all',
    sortBy: 'date'
  });

  // filters will persist across page reloads
  return (
    <FilterPanel filters={filters} onChange={setFilters} />
  );
}
```

usePermissions

Location: `src/hooks/usePermissions.js`

Check user permissions:

jsx

```jsx
import { usePermissions } from '@/hooks/usePermissions';

function EventActions({ event }) {
  const { can } = usePermissions();

  return (
    <div>
      {can('events:edit') && (
        <Button onClick={() => editEvent(event)}>Edit</Button>
      )}
      {can('events:delete') && (
        <Button onClick={() => deleteEvent(event)}>Delete</Button>
      )}
    </div>
  );
}
```

Routing & Navigation

Route Configuration

Location: `src/routes/AppRoutes.jsx`

jsx

```jsx
import { BrowserRouter, Routes, Route, Navigate } from 'react-router-dom';
import ProtectedRoute from './ProtectedRoute';
import PublicRoute from './PublicRoute';

// Auth pages
import Login from '@/pages/auth/Login';
import Register from '@/pages/auth/Register';
import ForgotPassword from '@/pages/auth/ForgotPassword';
import ResetPassword from '@/pages/auth/ResetPassword';

// Main pages
import Dashboard from '@/pages/Dashboard';
import EventsList from '@/pages/events/EventsList';
import EventDetail from '@/pages/events/EventDetail';
// ... more imports

function AppRoutes() {
  return (
    <BrowserRouter>
      <Routes>
        {/* Public routes */}
        <Route element={<PublicRoute />}>
          <Route path="/login" element={<Login />} />
          <Route path="/register" element={<Register />} />
          <Route path="/forgot-password" element={<ForgotPassword />} />
          <Route path="/reset-password/:token" element={<ResetPassword />} />
        </Route>

        {/* Protected routes */}
        <Route element={<ProtectedRoute />}>
          <Route path="/" element={<Navigate to="/dashboard" replace />} />
          <Route path="/dashboard" element={<Dashboard />} />

          {/* Events */}
          <Route path="/events" element={<EventsList />} />
          <Route path="/events/new" element={<EventForm />} />
          <Route path="/events/:id" element={<EventDetail />} />
          <Route path="/events/:id/edit" element={<EventForm />} />
          <Route path="/calendar" element={<EventCalendar />} />

          {/* Clients */}
          <Route path="/clients" element={<ClientsList />} />
          <Route path="/clients/new" element={<ClientForm />} />
          <Route path="/clients/:id" element={<ClientDetail />} />
          <Route path="/clients/:id/edit" element={<ClientForm />} />

          {/* Partners */}
          <Route path="/partners" element={<PartnersList />} />
          <Route path="/partners/new" element={<PartnerForm />} />
          <Route path="/partners/:id" element={<PartnerDetail />} />
          <Route path="/partners/:id/edit" element={<PartnerForm />} />

          {/* Payments */}
          <Route path="/payments" element={<PaymentsList />} />
          <Route path="/payments/new" element={<PaymentForm />} />
          <Route path="/payments/:id" element={<PaymentDetail />} />

          {/* Finance */}
          <Route path="/finance" element={<FinanceOverview />} />
          <Route path="/finance/reports" element={<FinanceReports />} />

          {/* Tasks */}
          <Route path="/tasks" element={<TasksList />} />
          <Route path="/tasks/board" element={<TaskBoard />} />
          <Route path="/tasks/:id" element={<TaskDetail />} />

          {/* Reminders */}
          <Route path="/reminders" element={<RemindersList />} />

          {/* Team */}
          <Route path="/team" element={<TeamList />} />
          <Route path="/team/invite" element={<InviteTeam />} />
          <Route path="/team/:id" element={<TeamMemberDetail />} />

          {/* Roles */}
          <Route path="/roles" element={<RolesList />} />
          <Route path="/roles/new" element={<RoleForm />} />
          <Route path="/roles/:id/edit" element={<RoleForm />} />

          {/* Settings */}
          <Route path="/settings" element={<VenueSettings />} />
          <Route path="/settings/profile" element={<ProfileSettings />} />
          <Route path="/settings/security" element={<SecuritySettings />} />
        </Route>

        {/* 404 */}
        <Route path="*" element={<NotFound />} />
      </Routes>
```

```
    </BrowserRouter>
  );
}
```

Protected Routes

Location: `src/routes/ProtectedRoute.jsx`

jsx

```jsx
import { Navigate, Outlet } from 'react-router-dom';
import { useAuth } from '@/context/AuthContext';
import MainLayout from '@/components/layout/MainLayout';
import LoadingSpinner from '@/components/common/LoadingSpinner';

function ProtectedRoute() {
  const { isAuthenticated, loading } = useAuth();

  if (loading) {
    return (
      <div className="flex items-center justify-center min-h-screen">
        <LoadingSpinner size="xl" />
      </div>
    );
  }

  if (!isAuthenticated) {
    return <Navigate to="/login" replace />;
  }

  return (
    <MainLayout>
      <Outlet />
    </MainLayout>
  );
}
```

Navigation

Use React Router's navigation hooks:

jsx

```jsx
import { useNavigate, useParams, useSearchParams } from 'react-router-dom';

function MyComponent() {
  const navigate = useNavigate();
  const { id } = useParams();
  const [searchParams, setSearchParams] = useSearchParams();

  // Navigate to a page
  const goToEvent = (eventId) => {
    navigate(`/events/${eventId}`);
  };

  // Navigate back
  const goBack = () => {
    navigate(-1);
  };

  // Update query params
  const updateFilters = (status) => {
    setSearchParams({ status });
  };

  return (
    <div>
      <Button onClick={() => goToEvent('123')}>View Event</Button>
      <Button onClick={goBack}>Go Back</Button>
    </div>
  );
}
```

Authentication & Authorization

Authentication Flow

```
1. User visits app
   ↓
2. Check for token in localStorage
   ↓
3a. Token exists → Validate with backend
   ↓
3b. No token → Redirect to /login
   ↓
4. User enters credentials
   ↓
5. POST /api/v1/auth/login
   ↓
6. Backend validates & returns token
   ↓
7. Store token & user in localStorage
   ↓
8. Redirect to /dashboard
```

Login Implementation

jsx

```
import { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import { useAuth } from '@/context/AuthContext';
import { toast } from 'react-hot-toast';

function Login() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [loading, setLoading] = useState(false);

  const { login } = useAuth();
  const navigate = useNavigate();

  const handleSubmit = async (e) => {
    e.preventDefault();
    setLoading(true);

    try {
      await login(email, password);
      toast.success('Welcome back!');
      navigate('/dashboard');
    } catch (error) {
      toast.error(error.message || 'Invalid credentials');
    } finally {
      setLoading(false);
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <Input
        label="Email"
        type="email"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
        required
      />
      <Input
        label="Password"
        type="password"
        value={password}
        onChange={(e) => setPassword(e.target.value)}
        required
      />
      <Button type="submit" loading={loading}>
        Sign In
      </Button>
    </form>
  );
}
```

Role-Based Access Control (RBAC)

Permission Structure:

javascript

```javascript
{
  module: 'events',
  actions: ['view', 'create', 'edit', 'delete']
}
```

Common Permissions:

- `events:view` - View events
- `events:create` - Create events
- `events:edit` - Edit events
- `events:delete` - Delete events
- `clients:*` - All client permissions
- `finance:view` - View financial data
- `team:manage` - Manage team members
- `settings:edit` - Edit venue settings

Checking Permissions:

jsx

```jsx
import { usePermissions } from '@/hooks/usePermissions';

function EventActions({ event }) {
  const { can, hasRole } = usePermissions();

  // Check specific permission
  if (!can('events:delete')) {
    return null;
  }

  // Check role
  if (hasRole('admin') || hasRole('owner')) {
    return <AdminActions />;
  }

  return (
    <Button onClick={() => deleteEvent(event)}>
      Delete
    </Button>
  );
}
```

Route-Level Protection:

jsx

```
import { Navigate } from 'react-router-dom';
import { usePermissions } from '@/hooks/usePermissions';

function ProtectedPage({ requiredPermission, children }) {
  const { can } = usePermissions();

  if (!can(requiredPermission)) {
    return <Navigate to="/dashboard" replace />;
  }

  return children;
}

// Usage
<Route
  path="/finance"
  element={
    <ProtectedPage requiredPermission="finance:view">
      <FinanceOverview />
    </ProtectedPage>
  }
/>
```

Data Fetching Patterns

Pattern 1: List Page with Filters

jsx

```
import { useState, useEffect } from 'react';
import { eventService } from '@/api/services';
import { toast } from 'react-hot-toast';

function EventsList() {
  const [events, setEvents] = useState([]);
  const [pagination, setPagination] = useState(null);
  const [loading, setLoading] = useState(true);
  const [filters, setFilters] = useState({
    status: 'all',
    search: '',
    page: 1,
    limit: 10
  });

  useEffect(() => {
    fetchEvents();
  }, [filters]);

  const fetchEvents = async () => {
    setLoading(true);

    try {
      const { events, pagination } = await eventService.getAll(filters);
      setEvents(events);
      setPagination(pagination);
    } catch (error) {
      toast.error(error.message || 'Failed to fetch events');
    } finally {
      setLoading(false);
    }
  };

  const handleFilterChange = (key, value) => {
    setFilters(prev => ({ ...prev, [key]: value, page: 1 }));
  };

  const handlePageChange = (page) => {
    setFilters(prev => ({ ...prev, page }));
  };

  if (loading) return <LoadingSpinner />;

  return (
    <div>
      <SearchBar
        value={filters.search}
        onChange={(value) => handleFilterChange('search', value)}
      />

      <FilterDropdown
        value={filters.status}
        onChange={(value) => handleFilterChange('status', value)}
        options={[
          { value: 'all', label: 'All Events' },
          { value: 'pending', label: 'Pending' },
          { value: 'confirmed', label: 'Confirmed' },
        ]}
      />

      <Table
        columns={columns}
        data={events}
        onRowClick={(event) => navigate(`/events/${event.id}`)}
      />

      <Pagination
        currentPage={filters.page}
        totalPages={pagination.pages}
        onPageChange={handlePageChange}
      />
    </div>
  );
}
```

Pattern 2: Detail Page

jsx

```
import { useState, useEffect } from 'react';
import { useParams, useNavigate } from 'react-router-dom';
import { eventService } from '@/api/services';
import { toast } from 'react-hot-toast';

function EventDetail() {
  const { id } = useParams();
  const navigate = useNavigate();
  const [event, setEvent] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    fetchEvent();
  }, [id]);

  const fetchEvent = async () => {
    setLoading(true);

    try {
      const { event } = await eventService.getById(id);
      setEvent(event);
    } catch (error) {
      if (error.status === 404) {
        toast.error('Event not found');
        navigate('/events');
      } else {
        toast.error(error.message);
      }
    } finally {
      setLoading(false);
    }
  };

  const handleDelete = async () => {
    if (!confirm('Delete this event?')) return;

    try {
      await eventService.delete(id);
      toast.success('Event deleted');
      navigate('/events');
    } catch (error) {
      toast.error(error.message);
    }
  };
  if (loading) return <LoadingSpinner />;
  if (!event) return <NotFound />;

  return (
    <div>
      <h1>{event.title}</h1>
      <p>{event.description}</p>

      <Button onClick={() => navigate(`/events/${id}/edit`)}>
        Edit
      </Button>
      <Button variant="danger" onClick={handleDelete}>
        Delete
      </Button>
    </div>
  );
}
```

Pattern 3: Create/Edit Form

jsx

```
import { useState, useEffect } from 'react';
import { useParams, useNavigate } from 'react-router-dom';
import { eventService } from '@/api/services';
import { toast } from 'react-hot-toast';

function EventForm() {
  const { id } = useParams();
  const navigate = useNavigate();
  const isEditing = Boolean(id);

  const [formData, setFormData] = useState({
    title: '',
    description: '',
    date: '',
    client: '',
    status: 'pending'
  });
  const [errors, setErrors] = useState({});
  const [loading, setLoading] = useState(false);

  useEffect(() => {
    if (isEditing) {
      fetchEvent();
    }
  }, [id]);

  const fetchEvent = async () => {
    try {
      const { event } = await eventService.getById(id);
      setFormData(event);
    } catch (error) {
      toast.error('Failed to load event');
      navigate('/events');
    }
  };

  const handleChange = (field, value) => {
    setFormData(prev => ({ ...prev, [field]: value }));
    // Clear error for this field
    if (errors[field]) {
      setErrors(prev => ({ ...prev, [field]: null }));
    }
  };
```

```jsx
const handleSubmit = async (e) => {
  e.preventDefault();
  setLoading(true);
  setErrors({});

  try {
    if (isEditing) {
      const { event } = await eventService.update(id, formData);
      toast.success('Event updated!');
      navigate(`/events/${event.id}`);
    } else {
      const { event } = await eventService.create(formData);
      toast.success('Event created!');
      navigate(`/events/${event.id}`);
    }
  } catch (error) {
    if (error.status === 422 && error.errors) {
      setErrors(error.errors);
      toast.error('Please fix the errors');
    } else {
      toast.error(error.message);
    }
  } finally {
    setLoading(false);
  }
};

return (
  <form onSubmit={handleSubmit}>
    <Input
      label="Event Title"
      value={formData.title}
      onChange={(e) => handleChange('title', e.target.value)}
      error={errors.title}
      required
    />

    <Input
      label="Description"
      type="textarea"
      value={formData.description}
      onChange={(e) => handleChange('description', e.target.value)}
      error={errors.description}
    />
```

```
      <Input
        label="Event Date"
        type="date"
        value={formData.date}
        onChange={(e) => handleChange('date', e.target.value)}
        error={errors.date}
        required
      />

      <div className="flex gap-2">
        <Button
          type="button"
          variant="outline"
          onClick={() => navigate('/events')}
        >
          Cancel
        </Button>
        <Button type="submit" loading={loading}>
          {isEditing ? 'Update' : 'Create'} Event
        </Button>
      </div>
    </form>
  );
}
```

Pattern 4: Parallel Data Fetching

jsx

```
import { useState, useEffect } from 'react';
import { dashboardService } from '@/api/services';

function Dashboard() {
  const [stats, setStats] = useState(null);
  const [events, setEvents] = useState([]);
  const [payments, setPayments] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    fetchDashboardData();
  }, []);

  const fetchDashboardData = async () => {
    setLoading(true);

    try {
      // Fetch all data in parallel
      const [statsData, eventsData, paymentsData] = await Promise.all([
        dashboardService.getStats(),
        dashboardService.getUpcomingEvents({ limit: 5 }),
        dashboardService.getRecentPayments({ limit: 5 })
      ]);

      setStats(statsData);
      setEvents(eventsData.events);
      setPayments(paymentsData.payments);
    } catch (error) {
      toast.error('Failed to load dashboard');
    } finally {
      setLoading(false);
    }
  };

  if (loading) return <LoadingSpinner />;

  return (
    <div>
      <StatsCards stats={stats} />
      <UpcomingEvents events={events} />
      <RecentPayments payments={payments} />
    </div>
    );
  }
```

Form Handling

Form Validation

Client-Side Validation:

jsx

```javascript
const validateEvent = (data) => {
  const errors = {};

  if (!data.title || data.title.trim().length === 0) {
    errors.title = 'Event title is required';
  }

  if (data.title && data.title.length > 100) {
    errors.title = 'Title must be less than 100 characters';
  }

  if (!data.date) {
    errors.date = 'Event date is required';
  }

  if (data.date && new Date(data.date) < new Date()) {
    errors.date = 'Event date must be in the future';
  }

  if (!data.client) {
    errors.client = 'Client is required';
  }

  if (data.capacity && data.capacity < 1) {
    errors.capacity = 'Capacity must be at least 1';
  }

  return errors;
};

// Usage in form
const handleSubmit = async (e) => {
  e.preventDefault();

  const validationErrors = validateEvent(formData);
  if (Object.keys(validationErrors).length > 0) {
    setErrors(validationErrors);
    return;
  }

  // Proceed with submission
```

```
    await submitForm();
  };
```

Form Utilities

Location: `src/utils/validators.js`

javascript

```
export const validators = {
  required: (value) => {
    return value && value.toString().trim().length > 0;
  },

  email: (value) => {
    const regex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    return regex.test(value);
  },

  phone: (value) => {
    const regex = /^[\d\s\-\+\(\)]+$/;
    return regex.test(value);
  },

  minLength: (value, length) => {
    return value && value.length >= length;
  },

  maxLength: (value, length) => {
    return value && value.length <= length;
  },

  url: (value) => {
    try {
      new URL(value);
      return true;
    } catch {
      return false;
    }
  },

  number: (value) => {
    return !isNaN(parseFloat(value)) && isFinite(value);
  },

  positiveNumber: (value) => {
    return validators.number(value) && parseFloat(value) > 0;
  },

  date: (value) => {
    return !isNaN(new Date(value).getTime());
  },
  futureDate: (value) => {
    return new Date(value) > new Date();
  },

  pastDate: (value) => {
    return new Date(value) < new Date();
  }
};

// Error messages
export const errorMessages = {
  required: 'This field is required',
  email: 'Please enter a valid email address',
  phone: 'Please enter a valid phone number',
  minLength: (length) => `Must be at least ${length} characters`,
  maxLength: (length) => `Must be no more than ${length} characters`,
  url: 'Please enter a valid URL',
  number: 'Please enter a valid number',
  positiveNumber: 'Must be a positive number',
  date: 'Please enter a valid date',
  futureDate: 'Date must be in the future',
  pastDate: 'Date must be in the past'
};
```

Form State Management

Using Custom Hook:

jsx

```
// src/hooks/useForm.js
import { useState } from 'react';

export const useForm = (initialValues, onSubmit, validate) => {
  const [values, setValues] = useState(initialValues);
  const [errors, setErrors] = useState({});
  const [touched, setTouched] = useState({});
  const [isSubmitting, setIsSubmitting] = useState(false);

  const handleChange = (name, value) => {
    setValues(prev => ({ ...prev, [name]: value }));

    // Clear error when user types
    if (errors[name]) {
      setErrors(prev => ({ ...prev, [name]: null }));
    }
  };

  const handleBlur = (name) => {
    setTouched(prev => ({ ...prev, [name]: true }));

    // Validate on blur
    if (validate) {
      const validationErrors = validate(values);
      if (validationErrors[name]) {
        setErrors(prev => ({ ...prev, [name]: validationErrors[name] }));
      }
    }
  };

  const handleSubmit = async (e) => {
    e.preventDefault();

    // Validate all fields
    if (validate) {
      const validationErrors = validate(values);
      setErrors(validationErrors);

      if (Object.keys(validationErrors).length > 0) {
        return;
      }
    }

    setIsSubmitting(true);
    try {
      await onSubmit(values);
    } catch (error) {
      if (error.status === 422 && error.errors) {
        setErrors(error.errors);
      }
    } finally {
      setIsSubmitting(false);
    }
  };

  const resetForm = () => {
    setValues(initialValues);
    setErrors({});
    setTouched({});
  };

  return {
    values,
    errors,
    touched,
    isSubmitting,
    handleChange,
    handleBlur,
    handleSubmit,
    resetForm,
    setValues,
    setErrors
  };
};

// Usage
function EventForm() {
  const navigate = useNavigate();

  const validate = (values) => {
    const errors = {};
    if (!values.title) errors.title = 'Title is required';
    if (!values.date) errors.date = 'Date is required';
    return errors;
  };

  const handleSubmit = async (values) => {
    const { event } = await eventService.create(values);
    toast.success('Event created!');
    navigate(`/events/${event.id}`);
  };

  const {
    values,
    errors,
    handleChange,
    handleBlur,
    handleSubmit: onSubmit,
    isSubmitting
  } = useForm(
    { title: '', description: '', date: '' },
    handleSubmit,
    validate
  );

  return (
    <form onSubmit={onSubmit}>
      <Input
        label="Title"
        value={values.title}
        onChange={(e) => handleChange('title', e.target.value)}
        onBlur={() => handleBlur('title')}
        error={errors.title}
      />

      <Button type="submit" loading={isSubmitting}>
        Create Event
      </Button>
    </form>
  );
}
```

Error Handling

Global Error Boundary

Location: `src/components/common/ErrorBoundary.jsx`

jsx

```
import React from 'react';
import { AlertTriangle } from 'lucide-react';
import Button from './Button';

class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false, error: null };
  }

  static getDerivedStateFromError(error) {
    return { hasError: true, error };
  }

  componentDidCatch(error, errorInfo) {
    console.error('Error caught by boundary:', error, errorInfo);

    // Log to error tracking service (Sentry, LogRocket, etc.)
    // logErrorToService(error, errorInfo);
  }

  handleReset = () => {
    this.setState({ hasError: false, error: null });
    window.location.href = '/dashboard';
  };

  render() {
    if (this.state.hasError) {
      return (
        <div className="min-h-screen flex items-center justify-center bg-gray-50">
          <div className="max-w-md w-full bg-white rounded-xl shadow-lg p-8 text-center">
            <div className="w-16 h-16 bg-red-50 rounded-full flex items-center justify-center mx-auto mb-4">
              <AlertTriangle className="w-8 h-8 text-red-500" />
            </div>

            <h1 className="text-2xl font-bold text-gray-900 mb-2">
              Something went wrong
            </h1>

            <p className="text-gray-600 mb-6">
              We're sorry for the inconvenience. Please try refreshing the page.
            </p>

            {process.env.NODE_ENV === 'development' && (
          <div className="bg-gray-50 rounded-lg p-4 mb-6 text-left">
            <p className="text-xs font-mono text-gray-700">
              {this.state.error?.toString()}
            </p>
          </div>
        )}

        <div className="flex gap-2 justify-center">
          <Button onClick={this.handleReset}>
            Go to Dashboard
          </Button>
          <Button
            variant="outline"
            onClick={() => window.location.reload()}
          >
            Refresh Page
          </Button>
        </div>
      </div>
    </div>
  );
}

return this.props.children;
}
}

export default ErrorBoundary;
```

Toast Notifications

Using react-hot-toast:

jsx

```jsx
// Setup in main.jsx
import { Toaster } from 'react-hot-toast';

function App() {
  return (
    <>
      <AppRoutes />
      <Toaster
        position="top-right"
        toastOptions={{
          duration: 4000,
          style: {
            background: '#363636',
            color: '#fff',
          },
          success: {
            duration: 3000,
            iconTheme: {
              primary: '#10B981',
              secondary: '#fff',
            },
          },
          error: {
            duration: 5000,
            iconTheme: {
              primary: '#EF4444',
              secondary: '#fff',
            },
          },
        }}
      />
    </>
  );
}

// Usage throughout app
import { toast } from 'react-hot-toast';

// Success
toast.success('Event created successfully!');

// Error
toast.error('Failed to create event');
```

```
// Loading
const toastId = toast.loading('Creating event...');
// Later
toast.success('Event created!', { id: toastId });

// Custom
toast.custom((t) => (
  <div className={`${t.visible ? 'animate-enter' : 'animate-leave'} bg-white shadow-lg rounded-lg p-4`}>
    <p>Custom notification</p>
  </div>
));

// Promise
toast.promise(
  eventService.create(data),
  {
    loading: 'Creating event...',
    success: 'Event created!',
    error: 'Failed to create event',
  }
);
```

API Error Handling

Centralized Error Handler:

javascript

```
// src/utils/errorHandler.js
import { toast } from 'react-hot-toast';

export const handleApiError = (error, customMessages = {}) => {
  // Network error
  if (!error.status) {
    toast.error('Network error. Please check your connection.');
    return;
  }

  // Use custom message if provided
  if (customMessages[error.status]) {
    toast.error(customMessages[error.status]);
    return;
  }

  // Default messages by status code
  switch (error.status) {
    case 400:
      toast.error(error.message || 'Invalid request');
      break;

    case 401:
      // Handled by axios interceptor (auto-logout)
      toast.error('Session expired. Please login again.');
      break;

    case 403:
      toast.error('You do not have permission to perform this action');
      break;

    case 404:
      toast.error('Resource not found');
      break;

    case 422:
      // Validation errors - handled separately in forms
      toast.error('Please check the form for errors');
      break;

    case 429:
      toast.error('Too many requests. Please try again later.');
      break;
    case 500:
    case 502:
    case 503:
      toast.error('Server error. Please try again later.');
      break;

    default:
      toast.error(error.message || 'An unexpected error occurred');
  }
};

// Usage
try {
  await eventService.create(data);
} catch (error) {
  handleApiError(error, {
    403: 'You need manager role to create events',
    404: 'Client not found'
  });
}
```

Form Error Display

jsx

```
function FormErrorSummary({ errors }) {
  const errorList = Object.entries(errors).filter(([_, value]) => value);

  if (errorList.length === 0) return null;

  return (
    <div className="bg-red-50 border border-red-200 rounded-lg p-4 mb-6">
      <div className="flex items-start">
        <AlertTriangle className="w-5 h-5 text-red-500 mt-0.5 mr-3" />
        <div className="flex-1">
          <h3 className="text-sm font-medium text-red-800 mb-2">
            Please fix the following errors:
          </h3>
          <ul className="list-disc list-inside text-sm text-red-700 space-y-1">
            {errorList.map(([field, message]) => (
              <li key={field}>{message}</li>
            ))}
          </ul>
        </div>
      </div>
    </div>
  );
}

// Usage in form
<form onSubmit={handleSubmit}>
  <FormErrorSummary errors={errors} />
  {/* form fields */}
</form>
```

Performance Optimization

Code Splitting

Route-based code splitting:

jsx

```jsx
import { lazy, Suspense } from 'react';
import LoadingSpinner from '@/components/common/LoadingSpinner';

// Lazy load page components
const Dashboard = lazy(() => import('@/pages/Dashboard'));
const EventsList = lazy(() => import('@/pages/events/EventsList'));
const EventDetail = lazy(() => import('@/pages/events/EventDetail'));

function AppRoutes() {
  return (
    <BrowserRouter>
      <Suspense fallback={<LoadingSpinner />}>
        <Routes>
          <Route path="/dashboard" element={<Dashboard />} />
          <Route path="/events" element={<EventsList />} />
          <Route path="/events/:id" element={<EventDetail />} />
        </Routes>
      </Suspense>
    </BrowserRouter>
  );
}
```

React.memo for Component Optimization

jsx

```
import { memo } from 'react';

// Expensive component that shouldn't re-render unless props change
const EventCard = memo(({ event, onClick }) => {
  console.log('EventCard rendered');

  return (
    <div onClick={() => onClick(event.id)}>
      <h3>{event.title}</h3>
      <p>{event.description}</p>
    </div>
  );
});

export default EventCard;
```

useCallback for Function Memoization

jsx

```jsx
import { useCallback } from 'react';

function EventsList() {
  const [events, setEvents] = useState([]);

  // Without useCallback, this creates new function on every render
  const handleEventClick = useCallback((eventId) => {
    navigate(`/events/${eventId}`);
  }, [navigate]); // Only recreate if navigate changes

  return (
    <div>
      {events.map(event => (
        <EventCard
          key={event.id}
          event={event}
          onClick={handleEventClick} // Same function reference
        />
      ))}
    </div>
  );
}
```

useMemo for Expensive Calculations

jsx

```
import { useMemo } from 'react';

function FinanceReport({ transactions }) {
  // Expensive calculation - only recalculate when transactions change
  const summary = useMemo(() => {
    console.log('Calculating summary...');

    return {
      totalIncome: transactions
        .filter(t => t.type === 'income')
        .reduce((sum, t) => sum + t.amount, 0),
      totalExpense: transactions
        .filter(t => t.type === 'expense')
        .reduce((sum, t) => sum + t.amount, 0),
      balance: transactions.reduce((sum, t) =>
        sum + (t.type === 'income' ? t.amount : -t.amount), 0
      )
    };
  }, [transactions]);

  return (
    <div>
      <p>Income: ${summary.totalIncome}</p>
      <p>Expense: ${summary.totalExpense}</p>
      <p>Balance: ${summary.balance}</p>
    </div>
  );
}
```

Image Optimization

jsx

```jsx
// Lazy load images
function EventImage({ src, alt }) {
  return (
    <img
      src={src}
      alt={alt}
      loading="lazy"
      className="w-full h-48 object-cover"
    />
  );
}

// Responsive images
function ResponsiveImage({ event }) {
  return (
    <picture>
      <source
        media="(min-width: 1024px)"
        srcSet={`${event.image}?w=800 1x, ${event.image}?w=1600 2x`}
      />
      <source
        media="(min-width: 640px)"
        srcSet={`${event.image}?w=400 1x, ${event.image}?w=800 2x`}
      />
      <img
        src={`${event.image}?w=300`}
        alt={event.title}
        loading="lazy"
      />
    </picture>
  );
}
```

Debouncing

jsx

```
import { useDebounce } from '@/hooks/useDebounce';

function SearchEvents() {
  const [query, setQuery] = useState('');
  const debouncedQuery = useDebounce(query, 500);

  useEffect(() => {
    if (debouncedQuery) {
      searchEvents(debouncedQuery);
    }
  }, [debouncedQuery]);

  return (
    <input
      value={query}
      onChange={(e) => setQuery(e.target.value)}
      placeholder="Search events..."
    />
  );
}
```

Virtual Scrolling (for large lists)

jsx

```
// Using react-window
import { FixedSizeList } from 'react-window';

function EventsVirtualList({ events }) {
  const Row = ({ index, style }) => (
    <div style={style}>
      <EventCard event={events[index]} />
    </div>
  );

  return (
    <FixedSizeList
      height={600}
      itemCount={events.length}
      itemSize={100}
      width="100%"
    >
      {Row}
    </FixedSizeList>
  );
}
```

Testing Strategy

Unit Testing with Jest

Setup:

bash

```bash
npm install --save-dev @testing-library/react @testing-library/jest-dom @testing-library/user-event vitest
```

Example Component Test:

javascript

```
// EventCard.test.jsx
import { render, screen, fireEvent } from '@testing-library/react';
import { describe, it, expect, vi } from 'vitest';
import EventCard from './EventCard';

describe('EventCard', () => {
  const mockEvent = {
    id: '1',
    title: 'Test Event',
    description: 'Test Description',
    date: '2024-12-01',
    status: 'confirmed'
  };

  it('renders event information', () => {
    render(<EventCard event={mockEvent} />);

    expect(screen.getByText('Test Event')).toBeInTheDocument();
    expect(screen.getByText('Test Description')).toBeInTheDocument();
  });

  it('calls onClick when clicked', () => {
    const handleClick = vi.fn();
    render(<EventCard event={mockEvent} onClick={handleClick} />);

    fireEvent.click(screen.getByText('Test Event'));

    expect(handleClick).toHaveBeenCalledWith('1');
  });

  it('displays correct status badge', () => {
    render(<EventCard event={mockEvent} />);

    const badge = screen.getByText('confirmed');
    expect(badge).toHaveClass('bg-green-100');
  });
});
```

Service Testing:

javascript

```
// eventService.test.js
import { describe, it, expect, vi, beforeEach } from 'vitest';
import { eventService } from '@/api/services';
import api from '@/api/axios';

vi.mock('@/api/axios');

describe('eventService', () => {
  beforeEach(() => {
    vi.clearAllMocks();
  });

  it('fetches all events', async () => {
    const mockEvents = [
      { id: '1', title: 'Event 1' },
      { id: '2', title: 'Event 2' }
    ];

    api.get.mockResolvedValue({
      data: { data: { events: mockEvents } }
    });

    const result = await eventService.getAll();

    expect(api.get).toHaveBeenCalledWith('/events', { params: {} });
    expect(result.events).toEqual(mockEvents);
  });

  it('creates an event', async () => {
    const newEvent = { title: 'New Event', date: '2024-12-01' };
    const createdEvent = { id: '1', ...newEvent };

    api.post.mockResolvedValue({
      data: { data: { event: createdEvent } }
    });

    const result = await eventService.create(newEvent);

    expect(api.post).toHaveBeenCalledWith('/events', newEvent);
    expect(result.event).toEqual(createdEvent);
  });
});
```

Integration Testing

javascript

```jsx
// EventsList.integration.test.jsx
import { render, screen, waitFor } from '@testing-library/react';
import { BrowserRouter } from 'react-router-dom';
import { describe, it, expect, vi } from 'vitest';
import EventsList from './EventsList';
import { eventService } from '@/api/services';

vi.mock('@/api/services');

describe('EventsList Integration', () => {
  it('fetches and displays events', async () => {
    const mockEvents = [
      { id: '1', title: 'Event 1', status: 'confirmed' },
      { id: '2', title: 'Event 2', status: 'pending' }
    ];

    eventService.getAll.mockResolvedValue({
      events: mockEvents,
      pagination: { page: 1, total: 2 }
    });

    render(
      <BrowserRouter>
        <EventsList />
      </BrowserRouter>
    );

    // Loading state
    expect(screen.getByRole('status')).toBeInTheDocument();

    // Wait for data
    await waitFor(() => {
      expect(screen.getByText('Event 1')).toBeInTheDocument();
      expect(screen.getByText('Event 2')).toBeInTheDocument();
    });
  });

  it('handles error state', async () => {
    eventService.getAll.mockRejectedValue({
      message: 'Failed to fetch events'
    });

    render(
      <BrowserRouter>
```

```
        <EventsList />
      </BrowserRouter>
    );

    await waitFor(() => {
      expect(screen.getByText(/failed to fetch/i)).toBeInTheDocument();
    });
  });
});
```

E2E Testing with Playwright (Optional)

javascript

```
// tests/e2e/events.spec.js
import { test, expect } from '@playwright/test';

test.describe('Events Management', () => {
  test.beforeEach(async ({ page }) => {
    // Login
    await page.goto('/login');
    await page.fill('[name="email"]', 'test@example.com');
    await page.fill('[name="password"]', 'password123');
    await page.click('button[type="submit"]');
    await page.waitForURL('/dashboard');
  });

  test('creates a new event', async ({ page }) => {
    await page.goto('/events');
    await page.click('text=Create Event');

    await page.fill('[name="title"]', 'Test Event');
    await page.fill('[name="description"]', 'Test Description');
    await page.fill('[name="date"]', '2024-12-01');

    await page.click('button:has-text("Create Event")');

    await expect(page).toHaveURL(/\/events\/\d+/);
    await expect(page.locator('text=Test Event')).toBeVisible();
  });

  test('filters events by status', async ({ page }) => {
    await page.goto('/events');

    await page.click('[data-testid="status-filter"]');
    await page.click('text=Confirmed');

    await expect(page.locator('[data-status="confirmed"]')).toBeVisible();
    await expect(page.locator('[data-status="pending"]')).not.toBeVisible();
  });
});
```

Deployment

Build for Production

bash

```bash
# Build the application
npm run build

# Output will be in dist/ folder
# dist/
# ├── assets/
# │   ├── index-[hash].js
# │   ├── index-[hash].css
# │   └── [other assets]
# └── index.html
```

Environment Configuration

Production `.env`:

env

```env
VITE_API_URL=https://api.yourdomain.com/api/v1
VITE_APP_NAME=Fiesta
VITE_ENABLE_ANALYTICS=true
VITE_ENABLE_DEBUG=false
```

Deployment Options

Option 1: Vercel

bash

```bash
# Install Vercel CLI
npm install -g vercel

# Deploy
vercel

# Production deployment
vercel --prod
```

vercel.json:

json

```json
{
  "buildCommand": "npm run build",
  "outputDirectory": "dist",
  "framework": "vite",
  "rewrites": [
    { "source": "/(.*)", "destination": "/index.html" }
  ]
}
```

Option 2: Netlify

bash

```bash
# Install Netlify CLI
npm install -g netlify-cli

# Deploy
netlify deploy

# Production deployment
netlify deploy --prod
```

netlify.toml:

toml

```
[build]
  command = "npm run build"
  publish = "dist"

[[redirects]]
  from = "/*"
  to = "/index.html"
  status = 200
```

Option 3: AWS S3 + CloudFront

bash

```bash
# Build
npm run build

# Upload to S3
aws s3 sync dist/ s3://your-bucket-name --delete

# Invalidate CloudFront cache
aws cloudfront create-invalidation \
  --distribution-id YOUR_DISTRIBUTION_ID \
  --paths "/*"
```

Option 4: Docker

Dockerfile:

dockerfile

```dockerfile
# Build stage
FROM node:18-alpine as build

WORKDIR /app

COPY package*.json ./
RUN npm ci

COPY . .
RUN npm run build

# Production stage
FROM nginx:alpine

COPY --from=build /app/dist /usr/share/nginx/html
COPY nginx.conf /etc/nginx/conf.d/default.conf

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

nginx.conf:

nginx

```
server {
    listen 80;
    server_name _;
    root /usr/share/nginx/html;
    index index.html;

    location / {
        try_files $uri $uri/ /index.html;
    }

    # Cache static assets
    location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg)$ {
        expires 1y;
        add_header Cache-Control "public, immutable";
    }

    # Security headers
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header X-XSS-Protection "1; mode=block" always;

    # Gzip compression
    gzip on;
    gzip_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss tex
}
```

Build and run:

bash

```
# Build image
docker build -t fiesta-frontend .

# Run container
docker run -p 80:80 fiesta-frontend
```

Performance Monitoring

Add analytics (Google Analytics example):

javascript

```
// src/utils/analytics.js
export const initAnalytics = () => {
  if (import.meta.env.VITE_ENABLE_ANALYTICS === 'true') {
    // Initialize GA4
    window.dataLayer = window.dataLayer || [];
    function gtag(){dataLayer.push(arguments);}
    gtag('js', new Date());
    gtag('config', 'G-XXXXXXXXXX');
  }
};

export const trackPageView = (path) => {
  if (window.gtag) {
    window.gtag('event', 'page_view', {
      page_path: path
    });
  }
};

export const trackEvent = (eventName, params = {}) => {
  if (window.gtag) {
    window.gtag('event', eventName, params);
  }
};

// Usage in App.jsx
import { useEffect } from 'react';
import { useLocation } from 'react-router-dom';
import { trackPageView } from '@/utils/analytics';

function App() {
  const location = useLocation();

  useEffect(() => {
    trackPageView(location.pathname);
  }, [location]);

  return <AppRoutes />;
}
```

Troubleshooting

Common Issues

1. "Module not found" errors

Problem: Import paths not resolving

Solution:

javascript

```javascript
// vite.config.js
import { defineConfig } from 'vite';
import react from '@vitejs/plugin-react';
import path from 'path';

export default defineConfig({
  plugins: [react()],
  resolve: {
    alias: {
      '@': path.resolve(__dirname, './src'),
    },
  },
});
```

2. API calls failing with CORS errors

Problem: Browser blocks API requests

Solution: Backend must include CORS headers:

javascript

```javascript
// Backend should send:
Access-Control-Allow-Origin: http://localhost:5173
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS
Access-Control-Allow-Headers: Content-Type, Authorization, X-Venue-ID
```

3. White screen after build

Problem: Assets not loading correctly

Solution: Check base URL in vite.config.js:

javascript

```javascript
export default defineConfig({
  base: '/', // or '/your-subdirectory/' if deployed to subdirectory
});
```

4. Dark mode not persisting

Problem: Theme resets on page refresh

Solution: Save to localStorage:

javascript

```javascript
// ThemeContext.jsx
useEffect(() => {
  const savedTheme = localStorage.getItem('theme') || 'light';
  setTheme(savedTheme);
  document.documentElement.classList.toggle('dark', savedTheme === 'dark');
}, []);

const toggleTheme = () => {
  const newTheme = theme === 'light' ? 'dark' : 'light';
  setTheme(newTheme);
  localStorage.setItem('theme', newTheme);
  document.documentElement.classList.toggle('dark', newTheme === 'dark');
};
```

5. Calendar not displaying correctly

Problem: React Calendar styles missing

Solution: Import CSS:

javascript

```javascript
// main.jsx or Calendar component
import 'react-calendar/dist/Calendar.css';
```

6. Token expiration not handled

Problem: User not redirected on 401

Solution: Check axios interceptor:

javascript

```javascript
// axios.js
api.interceptors.response.use(
  (response) => response,
  (error) => {
    if (error.response?.status === 401) {
      localStorage.clear();
      window.location.href = '/login';
    }
    return Promise.reject(error);
  }
);
```

Debug Mode

Enable detailed logging:

javascript

```javascript
// src/utils/logger.js
const isDev = import.meta.env.DEV;
const isDebug = import.meta.env.VITE_ENABLE_DEBUG === 'true';

export const logger = {
  log: (...args) => {
    if (isDev || isDebug) {
      console.log('📝', ...args);
    }
  },

  info: (...args) => {
    if (isDev || isDebug) {
      console.info('ℹ️', ...args);
    }
  },

  warn: (...args) => {
    if (isDev || isDebug) {
      console.warn('⚠️', ...args);
    }
  },

  error: (...args) => {
    console.error('❌', ...args);
    // Send to error tracking service in production
    if (!isDev) {
      // sendToErrorTracking(args);
    }
  },

  api: (method, url, data) => {
    if (isDev || isDebug) {
      console.group(`🌐 API ${method.toUpperCase()}: ${url}`);
      console.log('Data:', data);
      console.groupEnd();
    }
  }
};

// Usage
import { logger } from '@/utils/logger';
```

```
logger.api('GET', '/events', { status: 'active' });
logger.error('Failed to fetch events', error);
```

Best Practices

Code Organization

1. Component Structure:

jsx

```jsx
// ✅ Good
import { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import PropTypes from 'prop-types';

// External imports
import { CalendarIcon } from 'lucide-react';
import { toast } from 'react-hot-toast';

// Internal imports
import { eventService } from '@/api/services';
import Button from '@/components/common/Button';
import { formatDate } from '@/utils/formatters';

function EventCard({ event, onUpdate }) {
  // State
  const [loading, setLoading] = useState(false);

  // Hooks
  const navigate = useNavigate();

  // Effects
  useEffect(() => {
    // ...
  }, []);

  // Event handlers
  const handleClick = () => {
    navigate(`/events/${event.id}`);
  };

  // Render helpers
  const renderStatus = () => {
    // ...
  };

  // Main render
  return (
    <div onClick={handleClick}>
      {/* JSX */}
    </div>
  );
}
```

```
EventCard.propTypes = {
  event: PropTypes.object.isRequired,
  onUpdate: PropTypes.func
};

export default EventCard;
```

2. Naming Conventions:

javascript

```javascript
// ✅ Good naming
const usersList = []; // Arrays: plural
const user = {}; // Objects: singular
const isLoading = false; // Booleans: is/has/can prefix
const handleSubmit = () => {}; // Handlers: handle prefix
const fetchEvents = async () => {}; // Async: descriptive verb

// ❌ Avoid
const data = []; // Too generic
const flag = false; // Unclear purpose
const onClick = () => {}; // Missing handle prefix
const getData = async () => {}; // Not descriptive
```

3. Component Size:

jsx

```jsx
// ❌ Too large (>300 lines)
function EventForm() {
  // 500 lines of code...
}

// ✅ Break into smaller components
function EventForm() {
  return (
    <>
      <EventFormHeader />
      <EventFormBasicInfo />
      <EventFormDateTime />
      <EventFormPartners />
      <EventFormActions />
    </>
  );
}
```

4. Avoid Prop Drilling:

jsx

```jsx
// ❌ Prop drilling
<ParentComponent>
  <ChildComponent user={user}>
    <GrandchildComponent user={user}>
      <GreatGrandchildComponent user={user} />
    </GrandchildComponent>
  </ChildComponent>
</ParentComponent>

// ✅ Use Context
const UserContext = createContext();

<UserProvider value={user}>
  <ParentComponent>
    <ChildComponent>
      <GrandchildComponent>
        <GreatGrandchildComponent />
      </GrandchildComponent>
    </ChildComponent>
  </ParentComponent>
</UserProvider>

// In GreatGrandchildComponent
const user = useContext(UserContext);
```

Performance Best Practices

1. Avoid Inline Functions in JSX:

jsx

```
// ❌ Creates new function on every render
<Button onClick={() => handleClick(item.id)}>
  Click me
</Button>

// ✅ Use useCallback
const handleItemClick = useCallback(() => {
  handleClick(item.id);
}, [item.id]);

<Button onClick={handleItemClick}>
  Click me
</Button>
```

2. Memoize Expensive Calculations:

jsx

```
// ❌ Recalculates on every render
function EventsList({ events }) {
  const stats = calculateStats(events); // Expensive
  return <div>{stats.total}</div>;
}

// ✅ Use useMemo
function EventsList({ events }) {
  const stats = useMemo(() => calculateStats(events), [events]);
  return <div>{stats.total}</div>;
}
```

3. Lazy Load Images:

jsx

```
// ✅ Native lazy loading
<img src={event.image} alt={event.title} loading="lazy" />

// ✅ With intersection observer for more control
function LazyImage({ src, alt }) {
  const [imageSrc, setImageSrc] = useState(null);
  const imgRef = useRef();

  useEffect(() => {
    const observer = new IntersectionObserver(([entry]) => {
      if (entry.isIntersecting) {
        setImageSrc(src);
        observer.disconnect();
      }
    });

    if (imgRef.current) {
      observer.observe(imgRef.current);
    }

    return () => observer.disconnect();
  }, [src]);

  return <img ref={imgRef} src={imageSrc || placeholder} alt={alt} />;
}
```

Security Best Practices

1. Sanitize User Input:

javascript

```
import DOMPurify from 'dompurify';

// Render HTML safely
function EventDescription({ html }) {
  const sanitized = DOMPurify.sanitize(html);
  return <div dangerouslySetInnerHTML={{ __html: sanitized }} />;
}
```

2. Validate on Client AND Server:

javascript

```javascript
// Client-side validation (UX)
const validateEmail = (email) => {
  return /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email);
};

// But ALWAYS validate on server too (security)
// Never trust client-side validation alone
```

3. Avoid XSS:

jsx

```jsx
// ❌ Dangerous
<div dangerouslySetInnerHTML={{ __html: userInput }} />

// ✅ Safe
<div>{userInput}</div>

// ✅ If HTML is needed, sanitize first
<div dangerouslySetInnerHTML={{ __html: DOMPurify.sanitize(userInput) }} />
```

4. Secure Token Storage:

javascript

```javascript
// ⚠️ Current implementation (localStorage)
// Vulnerable to XSS attacks, but simpler
localStorage.setItem('token', token);

// ✅ Better: HttpOnly cookies (backend sets cookie)
// Not accessible to JavaScript, more secure
// Requires backend configuration
```

Accessibility Best Practices

1. Semantic HTML:

jsx

```jsx
// ❌ Non-semantic
<div onClick={handleClick}>Click me</div>

// ✅ Semantic
<button onClick={handleClick}>Click me</button>
```

2. ARIA Labels:

jsx

```jsx
// ✅ Add labels for screen readers
<button
  aria-label="Delete event"
  onClick={handleDelete}
>
  <TrashIcon />
</button>

<input
  type="text"
  aria-label="Search events"
  placeholder="Search..."
/>
```

3. Keyboard Navigation:

jsx

```jsx
function Modal({ isOpen, onClose }) {
  useEffect(() => {
    const handleEscape = (e) => {
      if (e.key === 'Escape') onClose();
    };

    if (isOpen) {
      document.addEventListener('keydown', handleEscape);
      return () => document.removeEventListener('keydown', handleEscape);
    }
  }, [isOpen, onClose]);

  return isOpen ? (
    <div role="dialog" aria-modal="true">
      {/* Modal content */}
    </div>
  ) : null;
}
```

4. Focus Management:

jsx

```jsx
function SearchBar() {
  const inputRef = useRef();

  useEffect(() => {
    // Focus input on mount
    inputRef.current?.focus();
  }, []);

  return (
    <input
      ref={inputRef}
      type="text"
      placeholder="Search..."
    />
  );
}
```

API Reference

Complete API Endpoints

Authentication

```
POST   /api/v1/auth/register          Register new venue
POST   /api/v1/auth/login             Login user
GET    /api/v1/auth/me                Get current user
PUT    /api/v1/auth/profile           Update profile
PUT    /api/v1/auth/change-password   Change password
POST   /api/v1/auth/forgot-password   Request password reset
POST   /api/v1/auth/reset-password    Reset password with token
POST   /api/v1/auth/logout            Logout user
```

Events

```
GET    /api/v1/events                 List all events
GET    /api/v1/events/stats           Get event statistics
GET    /api/v1/events/calendar        Get calendar view
POST   /api/v1/events                 Create event
GET    /api/v1/events/:id             Get single event
PUT    /api/v1/events/:id             Update event
PATCH  /api/v1/events/:id/status      Update event status
DELETE /api/v1/events/:id             Delete event
```

Clients

```
GET    /api/v1/clients                 List all clients
GET    /api/v1/clients/stats           Get client statistics
POST   /api/v1/clients                 Create client
GET    /api/v1/clients/:id             Get single client
PUT    /api/v1/clients/:id             Update client
DELETE /api/v1/clients/:id             Delete client
GET    /api/v1/clients/:id/events      Get client events
```

Partners

```
GET    /api/v1/partners                List all partners
GET    /api/v1/partners/stats          Get partner statistics
POST   /api/v1/partners                Create partner
GET    /api/v1/partners/:id            Get single partner
PUT    /api/v1/partners/:id            Update partner
DELETE /api/v1/partners/:id            Delete partner
GET    /api/v1/partners/:id/events     Get partner events
```

Payments

```
GET    /api/v1/payments                List all payments
GET    /api/v1/payments/stats          Get payment statistics
POST   /api/v1/payments                Create payment
GET    /api/v1/payments/:id            Get single payment
PUT    /api/v1/payments/:id            Update payment
DELETE /api/v1/payments/:id            Delete payment
POST   /api/v1/payments/:id/refund     Process refund
```

Finance

GET	/api/v1/finance	List all transactions
POST	/api/v1/finance	Create transaction
GET	/api/v1/finance/:id	Get single transaction
PUT	/api/v1/finance/:id	Update transaction
DELETE	/api/v1/finance/:id	Delete transaction
GET	/api/v1/finance/summary	Financial summary
GET	/api/v1/finance/cashflow	Cash flow report
GET	/api/v1/finance/expenses/breakdown	Expense breakdown
GET	/api/v1/finance/income/breakdown	Income breakdown
GET	/api/v1/finance/profit-loss	P&L statement
GET	/api/v1/finance/trends	Financial trends
GET	/api/v1/finance/tax-summary	Tax summary

Tasks

GET	/api/v1/tasks	List all tasks
GET	/api/v1/tasks/board	Kanban board view
GET	/api/v1/tasks/my	Current user's tasks
GET	/api/v1/tasks/stats	Task statistics
POST	/api/v1/tasks	Create task
GET	/api/v1/tasks/:id	Get single task
PUT	/api/v1/tasks/:id	Update task
DELETE	/api/v1/tasks/:id	Delete task
POST	/api/v1/tasks/:id/comments	Add comment
POST	/api/v1/tasks/:id/subtasks	Add subtask
PUT	/api/v1/tasks/:id/subtasks/:sid	Update subtask
DELETE	/api/v1/tasks/:id/subtasks/:sid	Delete subtask
POST	/api/v1/tasks/:id/attachments	Upload attachment
DELETE	/api/v1/tasks/:id/attachments/:aid	Delete attachment

Reminders

```
GET    /api/v1/reminders                List all reminders
GET    /api/v1/reminders/upcoming       Upcoming reminders
POST   /api/v1/reminders                Create reminder
GET    /api/v1/reminders/:id            Get single reminder
PUT    /api/v1/reminders/:id            Update reminder
DELETE /api/v1/reminders/:id            Delete reminder
POST   /api/v1/reminders/:id/snooze     Snooze reminder
```

Team

```
GET    /api/v1/team                     List team members
GET    /api/v1/team/stats               Team statistics
GET    /api/v1/team/invitations         List invitations
POST   /api/v1/team/invite              Invite team member
POST   /api/v1/team/accept-invitation   Accept invitation
POST   /api/v1/team/invitations/:id/resend Resend invitation
DELETE /api/v1/team/invitations/:id     Cancel invitation
GET    /api/v1/team/:id                 Get team member
PUT    /api/v1/team/:id                 Update team member
DELETE /api/v1/team/:id                 Remove team member
```

Roles

```
GET    /api/v1/roles                    List all roles
GET    /api/v1/roles/permissions        List all permissions
POST   /api/v1/roles                    Create role
GET    /api/v1/roles/:id                Get single role
PUT    /api/v1/roles/:id                Update role
DELETE /api/v1/roles/:id                Delete role
```

Venue

GET	/api/v1/venues/me	Get current venue
PUT	/api/v1/venues/me	Update venue
GET	/api/v1/venues/stats	Venue statistics
GET	/api/v1/venues/dashboard	Dashboard data
PUT	/api/v1/venues/subscription	Update subscription

Request/Response Examples

Create Event

✓

http

```
POST /api/v1/events
Content-Type: application/json
Authorization: Bearer <token>

{
  "title": "John & Jane Wedding",
  "description": "Beautiful wedding ceremony and reception",
  "eventType": "wedding",
  "date": "2024-12-15",
  "startTime": "17:00",
  "endTime": "23:00",
  "clientId": "client_123",
  "guestCount": 150,
  "venueFee": 5000,
  "status": "pending",
  "notes": "Client prefers round tables"
}

Response: 200 OK
{
  "success": true,
  "message": "Event created successfully",
  "data": {
    "event": {
      "id": "event_456",
      "title": "John & Jane Wedding",
      "description": "Beautiful wedding ceremony and reception",
      "eventType": "wedding",
      "date": "2024-12-15",
      "startTime": "17:00",
      "endTime": "23:00",
      "client": {
        "id": "client_123",
        "name": "John Doe",
        "email": "john@example.com"
      },
      "guestCount": 150,
      "venueFee": 5000,
      "status": "pending",
      "notes": "Client prefers round tables",
      "createdAt": "2024-11-01T10:00:00Z",
      "updatedAt": "2024-11-01T10:00:00Z"
    }
  }
}
```

List Events with Filters

http

```http
GET /api/v1/events?status=confirmed&page=1&limit=10&sortBy=date&order=asc
Authorization: Bearer <token>

Response: 200 OK
{
  "success": true,
  "data": {
    "events": [
      {
        "id": "event_123",
        "title": "Corporate Gala",
        "date": "2024-11-15",
        "status": "confirmed",
        "client": { "id": "client_456", "name": "ABC Corp" },
        "guestCount": 200
      },
      {
        "id": "event_124",
        "title": "Birthday Party",
        "date": "2024-11-20",
        "status": "confirmed",
        "client": { "id": "client_457", "name": "Jane Smith" },
        "guestCount": 50
      }
    ],
    "pagination": {
      "page": 1,
      "limit": 10,
      "total": 45,
      "pages": 5
    }
  }
}
```

Error Response

✓
http

```http
POST /api/v1/events
Content-Type: application/json
Authorization: Bearer <token>

{
  "title": "",
  "date": "2023-01-01"
}

Response: 422 Unprocessable Entity
{
  "success": false,
  "message": "Validation failed",
  "errors": {
    "title": "Event title is required",
    "date": "Event date must be in the future",
    "clientId": "Client is required"
  }
}
```

Utility Functions Reference

Date Formatters

Location: `src/utils/formatters.js`

✓
javascript

```
import { format, formatDistance, parseISO } from 'date-fns';

export const formatDate = (date, formatStr = 'MMM dd, yyyy') => {
  if (!date) return '';
  return format(parseISO(date), formatStr);
};

export const formatDateTime = (date) => {
  if (!date) return '';
  return format(parseISO(date), 'MMM dd, yyyy HH:mm');
};

export const formatTime = (time) => {
  if (!time) return '';
  return format(parseISO(`2000-01-01T${time}`), 'h:mm a');
};

export const formatRelativeTime = (date) => {
  if (!date) return '';
  return formatDistance(parseISO(date), new Date(), { addSuffix: true });
};

// Usage
formatDate('2024-11-15'); // "Nov 15, 2024"
formatDateTime('2024-11-15T14:30:00'); // "Nov 15, 2024 14:30"
formatTime('14:30'); // "2:30 PM"
formatRelativeTime('2024-11-01T10:00:00'); // "2 days ago"
```

Currency Formatters

javascript

```javascript
export const formatCurrency = (amount, currency = 'USD') => {
  return new Intl.NumberFormat('en-US', {
    style: 'currency',
    currency: currency,
  }).format(amount);
};

export const formatNumber = (number) => {
  return new Intl.NumberFormat('en-US').format(number);
};

// Usage
formatCurrency(5000); // "$5,000.00"
formatNumber(1234567); // "1,234,567"
```

Text Formatters

javascript

```javascript
export const truncate = (text, length = 50) => {
  if (!text || text.length <= length) return text;
  return `${text.substring(0, length)}...`;
};

export const capitalize = (text) => {
  if (!text) return '';
  return text.charAt(0).toUpperCase() + text.slice(1).toLowerCase();
};

export const titleCase = (text) => {
  if (!text) return '';
  return text
    .toLowerCase()
    .split(' ')
    .map(word => word.charAt(0).toUpperCase() + word.slice(1))
    .join(' ');
};

export const slugify = (text) => {
  return text
    .toLowerCase()
    .trim()
    .replace(/[^\w\s-]/g, '')
    .replace(/[\s_-]+/g, '-')
    .replace(/^-+|-+$/g, '');
};

// Usage
truncate('This is a very long text...', 20); // "This is a very long..."
capitalize('hello world'); // "Hello world"
titleCase('hello world'); // "Hello World"
slugify('Hello World!'); // "hello-world"
```

Constants

Location: `src/utils/constants.js`

javascript

```
export const EVENT_STATUSES = {
  DRAFT: 'draft',
  PENDING: 'pending',
  CONFIRMED: 'confirmed',
  IN_PROGRESS: 'in_progress',
  COMPLETED: 'completed',
  CANCELLED: 'cancelled'
};

export const EVENT_STATUS_COLORS = {
  [EVENT_STATUSES.DRAFT]: 'gray',
  [EVENT_STATUSES.PENDING]: 'yellow',
  [EVENT_STATUSES.CONFIRMED]: 'green',
  [EVENT_STATUSES.IN_PROGRESS]: 'blue',
  [EVENT_STATUSES.COMPLETED]: 'blue',
  [EVENT_STATUSES.CANCELLED]: 'red'
};

export const PAYMENT_STATUSES = {
  PENDING: 'pending',
  PAID: 'paid',
  PARTIALLY_PAID: 'partially_paid',
  OVERDUE: 'overdue',
  REFUNDED: 'refunded',
  CANCELLED: 'cancelled'
};

export const PAYMENT_METHODS = {
  CASH: 'cash',
  BANK_TRANSFER: 'bank_transfer',
  CREDIT_CARD: 'credit_card',
  CHECK: 'check',
  ONLINE: 'online'
};

export const USER_ROLES = {
  OWNER: 'owner',
  ADMIN: 'admin',
  MANAGER: 'manager',
  STAFF: 'staff'
};

export const TASK_PRIORITIES = {
  LOW: 'low',
  MEDIUM: 'medium',
  HIGH: 'high'
};

export const TASK_STATUSES = {
  TODO: 'todo',
  IN_PROGRESS: 'in_progress',
  DONE: 'done'
};
```

Appendix

Keyboard Shortcuts

Shortcut	Action
Cmd/Ctrl + K	Focus search bar
Cmd/Ctrl + N	Create new (context-dependent)
Cmd/Ctrl + S	Save form
Esc	Close modal/dropdown
?	Show keyboard shortcuts help

Browser Support

Browser	Minimum Version
Chrome	90+
Firefox	88+
Safari	14+
Edge	90+

Contributing Guidelines

Commit Message Format:

```
type(scope): subject

body

footer
```

Types:

- `feat`: New feature
- `fix`: Bug fix
- `docs`: Documentation only
- `style`: Code style (formatting)

- `refactor`: Code refactoring
- `test`: Adding tests
- `chore`: Maintenance tasks

Example:

```
feat(events): add calendar view

- Implemented monthly calendar view
- Added date navigation
- Integrated with react-calendar

Closes #123
```

Resources

Official Documentation:

- [React]
- [Vite]
- [Tailwind CSS]
- [React Router]

Community:

- GitHub Issues: Report bugs and request features
- Slack Channel: Join team discussions
- Documentation Site: Full guides and tutorials

Changelog

Version 2.0.0 - November 2024

- ✨ Added dark mode support
- ✨ Added calendar view for events
- ✨ Implemented advanced search and filtering
- ✨ Added toast notification system
- ✨ Implemented error boundary components
- 🔧 Consolidated API service layer
- 🔧 Improved performance with code splitting
- 🔧 Enhanced mobile responsiveness
- 🐛 Fixed token expiration handling
- 🐛 Fixed pagination issues

Version 1.0.0 - October 2024

- 🎉 Initial release
- ✨ Complete authentication system
- ✨ Event management CRUD
- ✨ Client management
- ✨ Partner management

- ✨ Payment tracking
- ✨ Dashboard with analytics
- ✨ Role-based access control

License

Document maintained by: Development Team
Last updated: November 2024
Version: 2.0

For questions or support, contact: support@fiesta.com