Fiesta Venue Management System - Frontend Documentation

🎨 Design Overview

The Fiesta Venue Management System features a modern, clean interface inspired by contemporary SaaS applications. The design emphasizes usability, clarity, and efficient workflow management for venue operators.

Design Principles

- **Modern & Clean**: Minimalist interface with ample white space
- **Orange Brand Identity**: Primary orange (🟧 `#F97316`) accent color throughout
- **Card-Based Layout**: Information organized in clean, elevated cards
- **Responsive Design**: Mobile-first approach with full desktop optimization
- **Intuitive Navigation**: Single-column sidebar with clear hierarchy

---

🏗️ Architecture

Technology Stack

- **Framework**: React 18+ with Hooks
- **Routing**: React Router DOM v6
- **Styling**: Tailwind CSS v3
- **Icons**: Lucide React
- **HTTP Client**: Axios
- **State Management**: React Context API
- **Build Tool**: Vite

Project Structure

```
src/
├── api/
│   └── services/
│       ├── api.js          # All API service endpoints
│       └── axios.js          # Axios instance configuration
├── components/
│   ├── common/              # Reusable UI components
│   │   ├── Button.jsx
│   │   ├── Input.jsx
│   │   ├── Modal.jsx
│   │   ├── Table.jsx
│   │   └── ...
│   └── layout/              # Layout components
│       ├── Sidebar.jsx       # Navigation sidebar
│       ├── TopBar.jsx        # Top navigation bar
│       └── MainLayout.jsx    # Main layout wrapper
├── context/
│   └── AuthContext.jsx       # Authentication state
├── pages/
│   ├── auth/                 # Authentication pages
│   ├── Dashboard.jsx         # Main dashboard
│   ├── events/               # Event management
│   ├── clients/              # Client management
│   ├── partners/             # Partner management
│   ├── payments/             # Payment tracking
│   ├── finance/              # Financial management
│   ├── tasks/                # Task management
│   ├── reminders/            # Reminder system
│   ├── team/                 # Team management
│   ├── roles/                # Role & permissions
│   └── settings/             # Application settings
└── routes/
    └── AppRoutes.jsx         # Route configuration
```

🎯 Core Components

1. Layout System

Sidebar Component

Location: `src/components/layout/Sidebar.jsx`

Features:

- Fixed left sidebar (264px width)
- Mobile responsive with slide-out drawer
- Section-based navigation (Overview, Operations, Financial, Management)
- Active route highlighting with orange background
- Collapsible on mobile with overlay backdrop
- Orange Fiesta logo with branding
- Sticky footer with copyright

Navigation Structure:

```
Overview
  └─ Dashboard

Operations
  ├─ Events
  ├─ Calendar
  ├─ Clients
  ├─ Partners
  ├─ Tasks
  └─ Reminders

Financial
  ├─ Payments
  ├─ Invoices
  └─ Finance

Management
  ├─ Team
  ├─ Roles
  └─ Settings
```

Key Props:

- `isOpen`: Boolean - Controls mobile sidebar visibility
- `onClose`: Function - Callback to close mobile sidebar

TopBar Component

Location: `src/components/layout/TopBar.jsx`

Features:

- Fixed top bar (64px height)
- Mobile hamburger menu button
- Notification bell with count badge
- User avatar with initials
- Responsive design (shows logo on mobile)
- Notification dropdown panel

Key Props:

- `onMenuClick`: Function - Opens mobile sidebar

MainLayout Component

Location: `src/components/layout/MainLayout.jsx`

Features:

- Wraps all authenticated pages
- Manages sidebar toggle state
- Responsive padding (smaller on mobile, larger on desktop)
- Gray background (`▢ #F9FAFB`)
- Proper spacing for fixed sidebar and topbar

---

2. Dashboard

Location: `src/pages/Dashboard.jsx`

Overview: The main landing page after login, displaying key metrics, upcoming events, and recent activity.

Layout Structure:

1. **Header Section**
    - Page title: "Dashboard"
    - Welcome message
    - Refresh button with loading animation
2. **Stats Grid** (4 cards)
    - Total Events
    - Revenue
    - Active Clients
    - Pending Payments

    Each card displays:
    - Title
    - Large value number
    - Colored icon in rounded background
    - Change indicator (green text)
3. **Content Grid** (2-column on large screens)
    - **Upcoming Events** (2/3 width)
        - Event cards with title, date, location, guest count
        - Status badges (confirmed/pending)
        - Hover effects
        - Empty state with icon
    - **Recent Activity** (1/3 width)
        - Payment activity feed
        - Activity icons
        - Time stamps
        - Badge indicators

API Integration:

- `dashboardService.getStats()` - Fetches dashboard statistics
- `dashboardService.getUpcomingEvents({ limit: 3 })` - Gets next 3 events
- `dashboardService.getRecentPayments({ limit: 3 })` - Gets latest payments

Key Features:

- Real-time data refresh
- Loading states with spinner
- Error handling with graceful fallbacks
- Empty states for no data
- Responsive grid layout
- Time-ago formatting ("2 hours ago")
- Currency formatting with commas

🎨 Design System

Color Palette

Primary Colors:

- Orange 500: 🟧 #F97316 - Primary brand color
- Orange 50: ⬜ #FFF7ED - Light background for active states

Semantic Colors:

- Green 500: 🟩 #10B981 - Success, revenue, positive metrics
- Green 50: ⬜ #ECFDF5 - Success backgrounds
- Yellow 500: 🟨 #F59E0B - Warnings, pending states
- Yellow 50: ⬜ #FFFBEB - Warning backgrounds
- Red 500: 🟥 #EF4444 - Errors, cancelled states
- Red 50: ⬜ #FEF2F2 - Error backgrounds
- Blue 500: 🟦 #3B82F6 - Information, completed states
- Blue 50: ⬜ #EFF6FF - Info backgrounds

Neutral Colors:

- Gray 900: `● #111827` - Primary text
- Gray 700: `● #374151` - Secondary text
- Gray 600: `● #4B5563` - Tertiary text
- Gray 500: `● #6B7280` - Placeholder text
- Gray 400: `● #9CA3AF` - Disabled text
- Gray 300: `● #D1D5DB` - Borders
- Gray 200: `● #E5E7EB` - Light borders
- Gray 100: `● #F3F4F6` - Background elements
- Gray 50: `● #F9FAFB` - Page background
- White: `● #FFFFFF` - Card backgrounds

Typography

Font Family: System font stack (default Tailwind)

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

Font Sizes:

- Display: `3xl` (30px) - Page titles
- Heading 1: `2xl` (24px) - Section titles
- Heading 2: `xl` (20px) - Card titles
- Heading 3: `lg` (18px) - Subsection titles
- Body Large: `base` (16px) - Primary text
- Body: `sm` (14px) - Secondary text
- Caption: `xs` (12px) - Labels, timestamps

Font Weights:

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

Spacing System

Following Tailwind's spacing scale (0.25rem = 4px increments):

- **Component Padding**:
  - Cards: `p-6` (24px)
  - Sidebar: `px-4` (16px)
  - Page container: `p-4 sm:p-6 lg:p-8`
- **Gap Between Elements**:
  - Tight: `gap-2` (8px)
  - Normal: `gap-4` (16px)
  - Comfortable: `gap-6` (24px)
- **Vertical Spacing**:
  - Sections: `space-y-6` (24px)
  - Cards: `space-y-4` (16px)
  - List items: `space-y-2` (8px)

Border Radius

- **Cards**: `rounded-xl` (12px)
- **Buttons**: `rounded-lg` (8px)
- **Badges**: `rounded-full` (999px)
- **Icons**: `rounded-lg` (8px) or `rounded-xl` (12px)

Shadows

- **Cards**: `border border-gray-200` (no shadow by default)
- **Hover**: `hover:shadow-md` (medium shadow on interaction)
- **Dropdowns**: `shadow-lg` (large shadow)
- **Modals**: `shadow-xl` (extra large shadow)

Icons

Library: Lucide React **Size**: Consistent `h-5 w-5` (20px) for navigation and inline icons **Large Icons**: `h-6 w-6` (24px) for stat cards **Empty States**: `h-12 w-12` (48px)

🔐 Authentication System

AuthContext

Location: `src/context/AuthContext.jsx`

Features:

- Manages authentication state globally
- Stores user data in localStorage
- Token management
- Automatic token refresh
- Protected route handling

State Properties:

```javascript
{
  user: Object | null,        // Current user data
  loading: Boolean,           // Auth loading state
  isAuthenticated: Boolean    // Auth status
}
```

Methods:

- `login(credentials)` - Authenticates user
- `logout()` - Clears session
- `updateUser(data)` - Updates user data

Protected Routes

Location: `src/routes/ProtectedRoute.jsx`

Behavior:

- Checks authentication status
- Redirects to login if not authenticated
- Shows loading spinner during auth check
- Preserves intended destination

📱 Responsive Design

Breakpoints (Tailwind defaults)

- **Mobile**: `< 640px` - Single column, mobile menu
- **Tablet**: `640px - 1024px` - Adjusted layouts, visible sidebar
- **Desktop**: `> 1024px` - Full layout with fixed sidebar

Mobile Optimizations

1. **Sidebar**:
   - Hidden by default
   - Slide-in drawer with backdrop
   - Hamburger menu in TopBar
2. **Dashboard**:
   - Stats: 1 column on mobile, 2 on tablet, 4 on desktop
   - Content: Stacked on mobile, side-by-side on desktop
   - Reduced padding on small screens
3. **Typography**:
   - Smaller headings on mobile
   - Responsive text sizing using `text-sm sm:text-base lg:text-lg`
4. **Touch Targets**:
   - Minimum 44x44px tap targets
   - Increased padding on mobile buttons
   - Larger menu items on mobile

🔌 API Integration

API Service Structure

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

All API services are organized by domain:

```javascript
// Dashboard
dashboardService.getStats()
dashboardService.getUpcomingEvents(params)
dashboardService.getRecentPayments(params)

// Events
eventService.getAll(params)
eventService.getById(id)
eventService.create(data)
eventService.update(id, data)
eventService.delete(id)

// Clients
clientService.getAll(params)
clientService.getById(id)
clientService.create(data)
clientService.update(id, data)

// ... and more
```

Axios Configuration

Location: `src/api/axios.js`

Features:

- Base URL configuration
- Automatic token injection
- Request/response interceptors
- Error handling
- Loading states

Request Flow:

```
Component → API Service → Axios → Backend API
   ↓                               ↓
Loading State                     Response/Error
   ↓                               ↓
Update UI ← Process Data ← Interceptor ←
```

⚡ Performance Optimizations

Code Splitting

- React.lazy() for route-based code splitting
- Suspense boundaries with loading fallbacks

API Optimization

- Promise.allSettled for parallel requests
- Request deduplication
- Loading state management

Image Optimization

- Lazy loading for images
- Responsive images with srcset
- WebP format support

Bundle Optimization

- Tree shaking enabled
- Production builds minified
- CSS purging with Tailwind

🧪 Development Guidelines

Component Creation

Best Practices:

1. Use functional components with hooks
2. Keep components under 300 lines
3. Extract reusable logic into custom hooks
4. Use PropTypes or TypeScript for type safety
5. Follow single responsibility principle

File Naming:

- PascalCase for components: `UserProfile.jsx`
- camelCase for utilities: `formatDate.js`
- kebab-case for styles: `custom-styles.css`

State Management

Local State: `useState` for component-specific state **Shared State**: Context API for auth, theme, etc. **Server State**: React Query (future enhancement)

Styling Guidelines

1. **Use Tailwind Utilities First**
   - Prefer utility classes over custom CSS
   - Use `@apply` sparingly in CSS files
2. **Consistent Spacing**
   - Follow 4px grid system
   - Use Tailwind spacing scale
3. **Color Usage**
   - Semantic color names from design system
   - No hardcoded hex values in components
4. **Responsive Design**
   - Mobile-first approach
   - Use `sm:`, `md:`, `lg:` prefixes

Code Organization

javascript

```javascript
// 1. Imports
import { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import { Icon } from 'lucide-react';
import { apiService } from '../api/services';

// 2. Component Definition
const MyComponent = ({ prop1, prop2 }) => {
  // 3. Hooks
  const navigate = useNavigate();
  const [state, setState] = useState(null);

  // 4. Effects
  useEffect(() => {
    // Effect logic
  }, []);

  // 5. Event Handlers
  const handleClick = () => {
    // Handler logic
  };

  // 6. Helper Functions
  const formatData = (data) => {
    // Helper logic
  };

  // 7. Conditional Rendering
  if (loading) return <Loader />;

  // 8. Main Render
  return (
    <div className="...">
      {/* JSX */}
    </div>
  );
};

// 9. Export
export default MyComponent;
```

🚀 Future Enhancements

Planned Features

1. **Advanced Dashboard**
    - Customizable widgets
    - Drag-and-drop layout
    - Chart.js integration for analytics
2. **Real-time Updates**
    - WebSocket integration
    - Live notifications
    - Real-time event updates
3. **Enhanced UX**
    - Skeleton loaders
    - Optimistic UI updates
    - Offline support with service workers
4. **Accessibility**
    - ARIA labels
    - Keyboard navigation
    - Screen reader support
5. **Internationalization**
    - Multi-language support
    - RTL layout support
    - Date/time localization
6. **Advanced Features**
    - Dark mode toggle
    - Custom themes
    - Export functionality (PDF, Excel)
    - Advanced search and filtering
    - Bulk operations

📝 Component Library

Common Components

All reusable components are located in `src/components/common/`:

- **Button**: Multiple variants (primary, secondary, outline, ghost)
- **Input**: Text inputs with icons and validation
- **Modal**: Customizable modal dialogs
- **Table**: Sortable, paginated data tables
- **Badge**: Status and category indicators
- **Card**: Container component for grouped content
- **EmptyState**: No-data placeholders with actions
- **LoadingSpinner**: Loading indicators
- **Pagination**: Page navigation component

🎯 Key User Flows

1. Login Flow

```
Login Page → Enter Credentials → API Call →
AuthContext Update → Redirect to Dashboard
```

2. Dashboard Flow

```
Dashboard Load → Fetch Stats (Parallel) →
Display Loading → Update UI → Enable Refresh
```

3. Event Management Flow

```
Events List → View Event → Edit Event →
Save Changes → Update List → Show Success
```

📊 Current Implementation Status

✅ Completed

- Authentication system (Login, Register, Forgot Password)
- Modern layout (Sidebar, TopBar, MainLayout)
- Dashboard with API integration
- Responsive design
- Navigation structure
- Design system implementation
- Orange brand identity

🚧 In Progress

- Event management CRUD
- Client management CRUD
- Partner management CRUD

📋 Planned

- Calendar view
- Payment tracking
- Financial analytics
- Task management
- Reminder system
- Team collaboration features
- Settings panels

🔧 Build & Deployment

Development

```bash
npm run dev
```

Starts Vite dev server on http://localhost:5173

Production Build

```bash
npm run build
```

Generates optimized production bundle in `dist/`

Preview Production

```bash
npm run preview
```

Preview production build locally

📚 Dependencies

Core

- `react`: ^18.2.0
- `react-dom`: ^18.2.0
- `react-router-dom`: ^6.x

UI & Styling

- `tailwindcss`: ^3.x
- `lucide-react`: Latest
- `autoprefixer`: ^10.x
- `postcss`: ^8.x

HTTP & Data

- `axios`: ^1.x

Development

- `vite`: ^5.x
- `@vitejs/plugin-react`: ^4.x
- `eslint`: ^8.x

💡 Tips & Tricks

Performance

- Use `React.memo()` for expensive components
- Implement `useCallback()` for event handlers passed as props
- Use `useMemo()` for expensive calculations

Debugging

- React DevTools for component inspection
- Network tab for API debugging
- Console logs with emojis for categorization

Code Quality

- Follow ESLint rules
- Use consistent formatting (Prettier)
- Write meaningful commit messages
- Keep components small and focused

🤝 Contributing

When contributing to the frontend:

1. Follow the established code structure
2. Match the design system (colors, spacing, typography)
3. Ensure mobile responsiveness
4. Test all user flows
5. Update this documentation for major changes

Document Version: 1.0
Last Updated: October 2025
Maintained By: Development Team