NEVER86 Restaurant Management System

Complete Project Documentation

Version: 2.0
Last Updated: 2024
Technology Stack: React 19, Vite, Tailwind CSS, Recharts

Table of Contents

Project Overview

NEVER86 is a comprehensive restaurant management system designed for modern restaurants. It provides role-based access for managers, servers, and kitchen staff, with features for order management, inventory tracking, staff scheduling, analytics, and more.

Key Capabilities

- **Multi-Role System:** Manager, Server, and Kitchen staff interfaces
- **Real-Time Order Management:** Table seating, order taking, kitchen tickets
- **Inventory Management:** Stock tracking, reorder alerts, usage predictions
- **Staff Scheduling:** Shift management, availability, time-off requests
- **Analytics & Reporting:** Comprehensive revenue, server, and menu performance metrics
- **Communication:** In-app chat system for staff coordination
- **Public Menu:** Customer-facing menu with QR code support

Technology Stack

- **Frontend Framework:** React 19.2.0
- **Build Tool:** Vite 7.3.1
- **Routing:** React Router DOM 7.12.0
- **Styling:** Tailwind CSS 3.4.1
- **Charts:** Recharts 3.6.0
- **Icons:** Lucide React 0.562.0
- **State Management:** React Context API

Architecture

High-Level Architecture

```
┌─────────────────────────────────────────────────────────┐
│                    Browser (Client)                      │
├─────────────────────────────────────────────────────────┤
│  React Application (NEVER86)                            │
│  ┌───────────────────────────────────────────────────┐  │
│  │  Context Providers (Auth, Data, Theme, Toast)     │  │
│  │  ┌─────────────────────────────────────────────┐  │  │
│  │  │  Router (React Router DOM)                  │  │  │
│  │  │  ┌───────────────────────────────────────┐  │  │  │
│  │  │  │  Protected Routes                     │  │  │  │
│  │  │  │  - Manager Pages                      │  │  │  │
│  │  │  │  - Server Pages                       │  │  │  │
│  │  │  │  - Kitchen Pages                      │  │  │  │
│  │  │  └───────────────────────────────────────┘  │  │  │
│  │  │  ┌───────────────────────────────────────┐  │  │  │
│  │  │  │  Public Routes                        │  │  │  │
│  │  │  │  - Login                              │  │  │  │
│  │  │  │  - Public Menu                        │  │  │  │
│  │  │  └───────────────────────────────────────┘  │  │  │
│  │  └─────────────────────────────────────────────┘  │  │
│  └───────────────────────────────────────────────────┘  │
├─────────────────────────────────────────────────────────┤
│  Data Persistence (localStorage)                         │
└─────────────────────────────────────────────────────────┘
```

Component Hierarchy

```
App
├── ErrorBoundary
│   └── BrowserRouter
│       └── ThemeProvider
│           └── ToastProvider
│               └── AuthProvider
│                   └── DataProvider
│                       └── Routes
│                           ├── Public Routes
│                           └── Protected Routes
│                               └── DashboardLayout
│                                   ├── Header
│                                   ├── Sidebar
│                                   └── Outlet (Page Content)
```

Data Flow

1. **User Action** → Component
2. **Component** → Context Function (e.g., `seatTable`, `createOrder`)
3. **Context** → Updates State
4. **State Change** → Triggers Re-render
5. **localStorage** → Auto-saves (debounced)

Features

Manager Features

Dashboard

- Real-time overview of restaurant operations
- Key performance indicators
- Recent activity feed
- Quick action buttons

Floor Plan

- Visual table layout
- Table status management (available, occupied, reserved)
- Section-based organization
- Drag-and-drop table assignment

Staff Management

- Staff roster
- Role assignment
- Performance tracking
- Availability management

Inventory Management

- Stock level tracking
- Low stock alerts
- Reorder queue
- Usage predictions
- Inventory history

Menu Management

- Menu item CRUD operations
- Category organization
- Price management
- Modifier groups
- Menu import/export

Sales & Analytics

- **Comprehensive Reporting Dashboard:**
  - Overview tab: High-level KPIs with trend indicators
  - Revenue tab: Detailed revenue analysis, day-of-week breakdowns
  - Servers tab: Performance metrics, leaderboards
  - Menu tab: Item performance, category analysis
  - Operations tab: Table turnover, efficiency metrics
- Custom date range selection
- Period-over-period comparisons
- CSV export functionality
- Multiple chart visualizations

Schedule Management

- Shift scheduling
- Staff availability
- Time-off requests
- Schedule conflicts detection

Profitability Analysis

- Menu item profitability
- Cost analysis
- Margin calculations
- Performance insights

Server Features

Dashboard

- Personal performance metrics
- Today's tables
- Quick stats
- Recent activity

Floor View

- Section-based table view
- My tables vs. all tables toggle
- Table status indicators
- Quick table access

Tables Management

- Table seating
- Order creation and modification
- Item additions/removals
- Special requests and notes
- Send to kitchen
- Bill out and payment processing

Stats & Competition

- Personal performance tracking
- Server leaderboard
- Competition rankings
- Tips tracking

Kitchen Features

Orders

- Kitchen ticket queue
- Order status management
- Item preparation tracking
- Ready notifications

Inventory

- Kitchen inventory view
- Stock levels
- Usage tracking

Shared Features

Chat System

- Role-based messaging
- Real-time notifications
- Message history
- Unread message counts

Settings

- Profile management
- Preferences
- Theme toggle (dark/light mode)
- Notification settings

Installation & Setup

Prerequisites

- Node.js 18+ and npm
- Modern web browser (Chrome, Firefox, Safari, Edge)

Installation Steps

1. **Clone or navigate to the project directory:**

   ```
   cd never86-app-v2
   ```

2. **Install dependencies:**

   ```
   npm install
   ```

3. **Start development server:**

   ```
   npm run dev
   ```

4. **Open in browser:**
   - Navigate to `http://localhost:5173` (or the port shown in terminal)

Build for Production

```
npm run build
```

The production build will be in the `dist/` directory.

Available Scripts

- `npm run dev` - Start development server
- `npm run build` - Build for production
- `npm run preview` - Preview production build
- `npm run lint` - Run ESLint

Project Structure

```
never86-app-v2/
├── public/                 # Static assets
├── src/
│   ├── assets/            # Images, icons
│   ├── components/        # Reusable components
│   │   ├── analytics/     # Analytics components
│   │   ├── inventory/     # Inventory components
│   │   ├── Layout/        # Layout components
│   │   ├── scheduling/    # Scheduling components
│   │   └── ui/            # UI primitives
│   ├── context/           # React Context providers
│   ├── data/              # Mock data and generators
│   ├── hooks/             # Custom React hooks
│   ├── lib/               # Utility libraries
│   │   ├── analytics/     # Analytics calculations
│   │   └── dataGeneration/# Data generation utilities
│   ├── pages/             # Page components
│   │   ├── kitchen/       # Kitchen role pages
│   │   ├── manager/       # Manager role pages
│   │   ├── menu/          # Public pages
│   │   └── server/        # Server role pages
│   ├── utils/             # Utility functions
│   ├── App.jsx            # Main app component
│   ├── main.jsx           # Application entry point
│   └── index.css          # Global styles
├── index.html             # HTML template
├── package.json           # Dependencies
├── tailwind.config.js     # Tailwind configuration
├── vite.config.js         # Vite configuration
└── README.md              # Project readme
```

Core Components

Layout Components

DashboardLayout

Location: `src/components/Layout/DashboardLayout.jsx`

Main layout wrapper for authenticated pages. Provides:

- Header with user info and notifications
- Sidebar navigation
- Content area for page components

Props: None (uses context for user data)

Header

Location: `src/components/Layout/Header.jsx`

Top navigation bar with:

- User profile menu
- Theme toggle
- Message center access

- Notifications

Sidebar

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

Navigation sidebar with:

- Role-based menu items
- Active route highlighting
- Collapsible design
- Unread message badges

UI Components

Card

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

Reusable card component with:

- Header, content, and description sections
- Consistent styling
- Dark mode support

Usage:

```
<Card>
  <CardHeader>
    <CardTitle>Title</CardTitle>
    <CardDescription>Description</CardDescription>
  </CardHeader>
  <CardContent>
    Content here
  </CardContent>
</Card>
```

Button

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

Styled button component with variants:

- `default`, `outline`, `ghost`, `destructive`

Badge

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

Badge component for labels with variants:

- `default`, `secondary`, `success`, `warning`, `destructive`

Input

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

Form input component with consistent styling.

Analytics Components

KPICard

Location: `src/components/analytics/KPICard.jsx`

Displays key performance indicators with:

- Icon
- Value
- Trend indicator (up/down/neutral)
- Subtitle

Props:

- `title` - KPI title
- `value` - Display value
- `icon` - Lucide icon component

- `trend` - 'up' | 'down' | 'neutral'
- `trendValue` - Percentage change
- `subtitle` - Additional info

RevenueChart

Location: `src/components/analytics/RevenueChart.jsx`

Area chart for revenue and tips trends.

Props:

- `data` - Array of daily sales data
- `title` - Chart title
- `height` - Chart height (default: 300)

ServerLeaderboard

Location: `src/components/analytics/ServerLeaderboard.jsx`

Displays server performance rankings.

Props:

- `servers` - Array of server objects
- `metric` - Metric to display (default: 'totalSales')
- `title` - Leaderboard title

Context Providers

AuthContext

Location: `src/context/AuthContext.jsx`

Manages user authentication state.

State:

- `currentUser` - Currently logged in user object

Functions:

- `login(email, password)` - Authenticate user
- `logout()` - Clear session

Usage:

```
const { currentUser, login, logout } = useAuth();
```

DataContext

Location: `src/context/DataContext.jsx`

Central state management for all restaurant data.

State:

- `tables` - Restaurant tables
- `orders` - Customer orders
- `menuItems` - Menu items
- `inventory` - Inventory stock
- `staff` - Staff members
- `chits` - Kitchen tickets
- `messages` - Chat messages
- And more...

Functions:

- `seatTable(tableId, guestCount)` - Seat guests at table
- `clearTable(tableId)` - Clear table after service
- `createOrder(tableId, items)` - Create new order
- `addItemToOrder(orderId, item)` - Add item to order
- `removeItemFromOrder(orderId, itemId)` - Remove item
- `sendToKitchen(orderId)` - Send order to kitchen
- And many more...

Usage:

```
const { tables, orders, seatTable, createOrder } = useData();
```

Persistence:

- All data auto-saves to localStorage
- Debounced saves (500ms delay)
- Version-based data migration

ThemeContext

Location: `src/context/ThemeContext.jsx`

Manages dark/light theme.

State:

- `isDarkMode` - Boolean for dark mode

Functions:

- `toggleDarkMode()` - Toggle theme

Usage:

```
const { isDarkMode, toggleDarkMode } = useTheme();
```

ToastContext

Location: `src/context/ToastContext.jsx`

Global notification system.

Functions:

- `showToast(message, type)` - Show toast
- `success(message)` - Success toast
- `error(message)` - Error toast
- `warning(message)` - Warning toast
- `info(message)` - Info toast

Usage:

```
const { showToast, success, error } = useToast();
```

Pages & Routes

Public Routes

`/login`

Component: `src/pages/Login.jsx`

User authentication page. Supports:

- Email/password login
- Role-based redirect after login
- Error handling

`/menu/:restaurantId?`

Component: `src/pages/menu/PublicMenu.jsx`

Public-facing menu page. Features:

- Category-based menu display
- QR code generation
- Responsive design

Manager Routes

All manager routes are prefixed with `/manager` and require manager role.

`/manager/dashboard`

Component: `src/pages/manager/Dashboard.jsx`

Manager dashboard with overview metrics.

`/manager/floor`

Component: `src/pages/manager/Floor.jsx`

Floor plan management with table layout.

`/manager/staff`

Component: `src/pages/manager/Staff.jsx`

Staff management and roster.

`/manager/inventory`

Component: `src/pages/manager/Inventory.jsx`

Inventory management with stock tracking.

`/manager/menu`

Component: `src/pages/manager/Menu.jsx`

Menu item management.

`/manager/sales`

Component: `src/pages/manager/Sales.jsx`

Sales overview and trends.

`/manager/analytics`

Component: `src/pages/manager/Analytics.jsx`

Comprehensive Analytics Dashboard with:

- **Overview Tab:** KPIs, revenue trends, server leaderboard, peak hours
- **Revenue Tab:** Revenue by day of week, daily trends, detailed breakdowns
- **Servers Tab:** Server leaderboard, performance charts, detailed stats
- **Menu Tab:** Category performance, top items, menu item breakdown
- **Operations Tab:** Table turnover, efficiency metrics, peak hours

Features:

- Custom date range selection
- Period-over-period comparisons
- CSV export
- Multiple chart types (line, bar, pie)

`/manager/schedule`

Component: `src/pages/manager/Schedule.jsx`

Staff scheduling interface.

`/manager/profitability`

Component: `src/pages/manager/Profitability.jsx`

Menu profitability analysis.

`/manager/chat`

Component: `src/pages/manager/Chat.jsx`

Manager chat interface.

`/manager/settings`

Component: `src/pages/manager/Settings.jsx`

Manager settings and preferences.

Server Routes

All server routes are prefixed with `/server` and require server role.

`/server/dashboard`

Component: `src/pages/server/Dashboard.jsx`

Server personal dashboard.

`/server/floor`

Component: `src/pages/server/Floor.jsx`

Floor view with section filtering.

`/server/tables`

Component: `src/pages/server/Tables.jsx`

Table and order management for servers.

`/server/stats`

Component: `src/pages/server/Stats.jsx`

Personal performance statistics.

`/server/competition`

Component: `src/pages/server/Competition.jsx`

Server competition leaderboard.

`/server/chat`

Component: `src/pages/server/Chat.jsx`

Server chat interface.

`/server/settings`

Component: `src/pages/server/Settings.jsx`

Server settings.

Kitchen Routes

All kitchen routes are prefixed with `/kitchen` and require kitchen role.

`/kitchen/orders`

Component: `src/pages/kitchen/Orders.jsx`

Kitchen ticket queue.

`/kitchen/inventory`

Component: `src/pages/kitchen/Inventory.jsx`

Kitchen inventory view.

`/kitchen/chat`

Component: `src/pages/kitchen/Chat.jsx`

Kitchen chat interface.

`/kitchen/settings`

Component: `src/pages/kitchen/Settings.jsx`

Kitchen settings.

Analytics & Reporting

Analytics Calculator

Location: `src/lib/analytics/kpiCalculator.js`

Core analytics functions:

`calculateServerLeaderboard(orders, staff)`

Calculates server performance metrics.

Returns: Array of server objects with:

- `id`, `name`, `revenue`, `tables`, `tips`, `rank`

`calculatePeakHours(orders)`

Calculates order distribution by hour.

Returns: Array of hour objects with:

- `hour` (0-23), `count`

`calculateRevenueMetrics(orders, sales, startDate, endDate, prevStartDate?, prevEndDate?)`

Comprehensive revenue analysis with period comparison.

Returns: Object with:

- Current period metrics (revenue, tips, tables, avg check, etc.)
- Daily averages
- Revenue by day of week
- Previous period comparison (if provided)

`calculateMenuPerformance(orders, menuItems, startDate, endDate)`

Menu item performance analysis.

Returns: Array of menu items sorted by quantity sold with:

- `name`, `category`, `price`, `quantitySold`, `totalRevenue`, `avgPrice`

`calculateHourlyRevenue(orders, startDate, endDate)`

Hourly revenue distribution.

Returns: Array of hour objects with:

- `hour`, `revenue`, `orders`, `tables`, `tips`, `avgOrderValue`, `label`

Analytics Dashboard Features

Date Range Selection

- Preset ranges: 7, 30, 60, 84 days
- Custom date range picker
- Period-over-period comparison toggle

Export Functionality

- CSV export with all metrics
- Includes KPIs, server stats, menu performance
- Period comparisons when enabled

Visualizations

- **Line Charts:** Revenue trends over time
- **Bar Charts:** Day-of-week analysis, server performance, menu items
- **Pie Charts:** Category revenue distribution
- **Area Charts:** Revenue and tips trends
- **Bar Visualizations:** Peak hours analysis

Data Management

Data Structures

Table Object

```
{
  id: string,
  number: number,
  section: string,
  seats: number,
  status: 'available' | 'occupied' | 'reserved',
  serverId: string | null,
  currentOrderId: string | null,
  seatedAt: string | null,
  guestCount: number
}
```

Order Object

```
{
  id: string,
  tableId: string,
  serverId: string,
  guestCount: number,
  status: 'pending' | 'sent' | 'preparing' | 'ready' | 'completed',
  items: Array<OrderItem>,
  createdAt: string,
  closedAt: string | null,
  tip: number,
  total: number
}
```

Menu Item Object

```
{
  id: string,
  name: string,
  description: string,
  price: number,
  category: string,
  modifiers: Array<ModifierGroup>,
  available: boolean,
  imageUrl: string | null
}
```

Data Persistence

- **Storage:** Browser localStorage
- **Auto-save:** Debounced (500ms delay)
- **Versioning:** Data version system for migrations
- **Key:** `never86_data_v{VERSION}`

Mock Data

Mock data is located in:

- `src/data/mockDataExtended.js` - Extended mock data
- `src/data/mockData.js` - Basic mock data
- `src/data/generatedHistoricalData.js` - Generated historical data

Styling & Theming

Tailwind CSS

The project uses Tailwind CSS for styling with:

- Custom color palette
- Dark mode support
- Responsive design utilities
- Custom component classes

Theme Configuration

File: `tailwind.config.js`

Custom colors:

- `primary` - Main brand color
- `secondary` - Secondary color
- `muted` - Muted text/background
- `brand.navy`, `brand.blue`, `brand.light`
- Role-specific colors: `manager.*`, `server.*`, `kitchen.*`

Dark Mode

Dark mode is managed by:

- `ThemeContext` for state
- `dark` class on HTML element
- CSS variables in `src/index.css`

Toggle via:

- Header theme toggle button
- System preference detection
- localStorage persistence

Development Guide

Adding a New Page

1. Create component in appropriate directory:

   ```
   src/pages/{role}/{PageName}.jsx
   ```

2. Add route in `src/App.jsx`:

   ```
   <Route path="/{role}/{page}" element={<PageName />} />
   ```

3. Add navigation item in `src/components/Layout/Sidebar.jsx`:

   ```
   { to: '/{role}/{page}', icon: IconName, label: 'Page Name' }
   ```

Adding a New Context

1. Create context file:

   ```
    // src/context/NewContext.jsx
   import { createContext, useContext, useState } from 'react';

   const NewContext = createContext();

   export function NewProvider({ children }) {
     const [state, setState] = useState(initialValue);
     // ... logic
     return (
       <NewContext.Provider value={{ state, setState }}>
         {children}
       </NewContext.Provider>
     );
   }

   export function useNew() {
     return useContext(NewContext);
   }
   ```

2. Add provider to `src/App.jsx`:

```
<NewProvider>
  {/* existing providers */}
</NewProvider>
```

Adding Analytics Functions

1. Add function to `src/lib/analytics/kpiCalculator.js`:

   ```
   export function calculateNewMetric(data) {
     // calculation logic
     return result;
   }
   ```

2. Export from `src/lib/analytics/index.js`:

   ```
   export * from './kpiCalculator.js';
   ```

Code Style

- Use functional components with hooks
- Use descriptive variable names
- Add JSDoc comments for complex functions
- Follow existing component patterns
- Use Tailwind classes for styling

API Reference

DataContext API

Table Management

```
seatTable(tableId, guestCount, serverId?)
clearTable(tableId)
updateTableStatus(tableId, status)
```

Order Management

```
createOrder(tableId, items)
addItemToOrder(orderId, item)
removeItemFromOrder(orderId, itemId)
updateOrderItem(orderId, itemId, updates)
sendToKitchen(orderId)
completeOrder(orderId)
```

Menu Management

```
addMenuItem(item)
updateMenuItem(itemId, updates)
deleteMenuItem(itemId)
```

Inventory Management

```
updateInventory(itemId, quantity)
addInventoryItem(item)
```

Staff Management

```
addStaffMember(member)
updateStaffMember(memberId, updates)
deleteStaffMember(memberId)
```

Messaging

```
sendMessage(message)
markMessageRead(messageId)
getUnreadMessageCount()
```

Analytics API

All analytics functions are exported from `src/lib/analytics/kpiCalculator.js`:

```
import {
  calculateServerLeaderboard,
  calculatePeakHours,
  calculateRevenueMetrics,
  calculateMenuPerformance,
  calculateHourlyRevenue
} from '../../lib/analytics/kpiCalculator';
```

Troubleshooting

Common Issues

App Won't Start

1. Check Node.js version (18+ required)
2. Delete `node_modules` and `package-lock.json`
3. Run `npm install` again
4. Check for console errors

Data Not Persisting

1. Check browser localStorage is enabled
2. Check data version in DataContext
3. Clear localStorage and refresh

Styling Issues

1. Ensure Tailwind is properly configured
2. Check `tailwind.config.js` content paths
3. Verify CSS imports in `main.jsx`

Build Errors

1. Check for syntax errors
2. Verify all imports are correct
3. Check for missing dependencies
4. Run `npm run lint` for linting errors

Debugging Tips

1. **React DevTools:** Install browser extension
2. **Console Logging:** Use `console.log` for debugging
3. **Network Tab:** Check for failed requests
4. **Application Tab:** Inspect localStorage data

Additional Resources

Documentation Files

- `README.md` - Project overview
- `MENU_IMPORT_GUIDE.md` - Menu import instructions
- `MENU_IMPORT_QUICK_REFERENCE.md` - Quick reference
- `MENU_IMPORT_TEMPLATE.js` - Import template

External Resources

- React Documentation
- Vite Documentation
- Tailwind CSS Documentation
- React Router Documentation
- Recharts Documentation

Version History

Version 2.0 (Current)

- Comprehensive analytics dashboard revamp
- Enhanced reporting with multiple tabs
- Period-over-period comparisons
- CSV export functionality
- Improved data safety checks
- Better error handling

Version 1.0

- Initial release
- Basic role-based system
- Core features implementation

Support & Contact

For issues, questions, or contributions, please refer to the project repository.

End of Documentation