Wine Store - Complete Project Documentation

Table of Contents

1. Project Overview
2. Getting Started
3. Project Structure
4. Technology Stack
5. Core Components
6. API Routes
7. Database Schema
8. Authentication
9. Styling & UI
10. State Management
11. File-by-File Breakdown
12. Development Workflow
13. Deployment
14. Troubleshooting

Project Overview

The Wine Store is a full-stack e-commerce application built with Next.js 15, featuring:

- **User Authentication** via Clerk
- **Product Management** with Prisma ORM
- **Shopping Cart** functionality
- **Payment Processing** via Stripe
- **Responsive Design** with Tailwind CSS
- **Admin Dashboard** for product management
- **Order Management** system
- **Favorites & Reviews** system

Getting Started

Prerequisites

- Node.js 18+
- npm or yarn
- PostgreSQL database
- Clerk account for authentication
- Stripe account for payments

Installation Steps

1. **Clone the repository**

```
git clone <your-repo-url>
cd store-wine
```

2. **Install dependencies**

```
npm install
```

3. **Environment Setup** Create a `.env.local` file with:

```
# Database
DATABASE_URL="postgresql://username:password@localhost:5432/wine_store"

# Clerk Authentication
NEXT_PUBLIC_CLERK_PUBLISHABLE_KEY=pk_test_...
CLERK_SECRET_KEY=sk_test_...

# Stripe
STRIPE_SECRET_KEY=sk_test_...
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_test_...

# App
NEXTAUTH_SECRET=your-secret-here
NEXTAUTH_URL=http://localhost:3000
```

4. **Database Setup**

```
# Generate Prisma client
npx prisma generate

# Run migrations
npx prisma migrate dev

# Seed the database
npm run seed
```

5. **Start Development Server**

```
npm run dev
```

The application will be available at `http://localhost:3000`

Project Structure

```
store-wine/
├── app/                     # Next.js 13+ App Router
│   ├── api/                # API routes
│   ├── admin/              # Admin dashboard
│   ├── cart/               # Shopping cart pages
│   ├── checkout/           # Checkout process
│   ├── favorites/          # User favorites
│   ├── orders/             # Order management
│   ├── products/           # Product pages
│   ├── reviews/            # Review system
│   ├── sign-in/            # Authentication
│   ├── sign-up/            # User registration
│   ├── globals.css         # Global styles
│   ├── layout.tsx          # Root layout
│   └── page.tsx            # Home page
├── components/              # React components
│   ├── ui/                 # Reusable UI components
│   ├── auth/               # Authentication components
│   ├── cart/               # Cart-related components
│   ├── form/               # Form components
│   ├── global/             # Global components
│   ├── home/               # Home page components
│   ├── navbar/             # Navigation components
│   ├── products/           # Product components
│   └── single-product/     # Single product components
│
```

```
├── lib/                    # Utility libraries
├── prisma/                 # Database schema & migrations
├── public/                 # Static assets
├── styles/                 # Additional styles
├── utils/                  # Utility functions
└── middleware.ts           # Next.js middleware
```

Technology Stack

Frontend

- **Next.js 15** - React framework with App Router
- **React 19** - UI library
- **TypeScript** - Type safety
- **Tailwind CSS 4** - Utility-first CSS framework

Backend

- **Next.js API Routes** - Server-side API endpoints
- **Prisma** - Database ORM
- **PostgreSQL** - Primary database

Authentication & Payments

- **Clerk** - User authentication & management
- **Stripe** - Payment processing

UI Components

- **shadcn/ui** - Pre-built component library
- **Radix UI** - Headless UI primitives
- **Lucide React** - Icon library

Development Tools

- **ESLint** - Code linting
- **PostCSS** - CSS processing
- **Turbopack** - Fast bundler

Core Components

1. Layout & Navigation

- **Root Layout** (`app/layout.tsx`) - Main application wrapper
- **Navbar** (`components/navbar/`) - Main navigation
- **Container** (`components/global/Container.tsx`) - Layout wrapper

2. Product Management

- **ProductsGrid** (`components/products/ProductsGrid.tsx`) - Product display
- **ProductCard** - Individual product component
- **FilterSidebar** - Product filtering
- **ProductsContainer** - Product listing wrapper

3. Shopping Cart

- **CartItemsList** (`components/cart/CartItemsList.tsx`) - Cart items display
- **CartTotals** - Price calculations
- **AddToCart** - Add products to cart

4. Authentication

- **CustomSignIn** (`components/auth/CustomSignIn.tsx`) - Sign-in form
- **UserButtonWrapper** - User menu wrapper
- **UserMenu** - User dropdown menu

5. Admin Dashboard

- **Admin Page** (`app/admin/page.tsx`) - Admin overview
- **Sales Page** (`app/admin/sales/page.tsx`) - Sales analytics

API Routes

Product Management

- `GET /api/products` - List all products
- `GET /api/products/[handle]` - Get single product
- `POST /api/products` - Create product (admin)

Cart & Orders

- `GET /api/cart-items` - Get cart items
- `POST /api/cart-items` - Add to cart
- `GET /api/orders` - List orders
- `POST /api/orders` - Create order

Payment

- `POST /api/payment` - Process payment
- `POST /api/confirm` - Confirm payment

Carousel

- `GET /api/carousel` - Get carousel data

Database Schema

The project uses Prisma with PostgreSQL. Key models include:

- **User** - User accounts and profiles
- **Product** - Wine products with details
- **Order** - Customer orders
- **OrderItem** - Individual items in orders
- **Review** - Product reviews
- **Favorite** - User favorites

Authentication

Authentication is handled by Clerk:

- User registration and login
- Protected routes
- User profile management
- Role-based access control

Styling & UI

- **Tailwind CSS 4** for utility-first styling
- **shadcn/ui** components for consistent design
- **Dark mode** support
- **Responsive design** for all screen sizes
- **Custom CSS variables** for theming

State Management

- **React hooks** for local state
- **Server components** for data fetching
- **Client components** for interactivity
- **Context API** for global state when needed

Development Workflow

1. **Feature Development**
    - Create feature branch
    - Implement changes
    - Test functionality
    - Create pull request
2. **Database Changes**
    - Update Prisma schema
    - Generate migration
    - Test locally
    - Deploy migration
3. **Component Development**
    - Create component file
    - Add TypeScript interfaces
    - Implement functionality
    - Add to storybook (if applicable)

Deployment

Vercel (Recommended)

1. Connect GitHub repository
2. Set environment variables
3. Deploy automatically on push

Other Platforms

- **Netlify** - Static site hosting
- **Railway** - Full-stack hosting
- **DigitalOcean** - VPS hosting

Troubleshooting

Common Issues

1. **Database Connection**
    - Check DATABASE_URL in .env
    - Ensure PostgreSQL is running
    - Run `npx prisma generate`
2. **Authentication Issues**
    - Verify Clerk keys
    - Check environment variables
    - Clear browser cache
3. **Build Errors**
    - Clear .next folder
    - Reinstall dependencies
    - Check TypeScript errors

4. **Styling Issues**
    - Verify Tailwind configuration
    - Check PostCSS setup
    - Clear CSS cache

Getting Help

- Check the GitHub issues
- Review the codebase
- Consult the documentation
- Reach out to the development team

This documentation provides a comprehensive overview of the Wine Store project. For specific implementation details, refer to the individual component files and their inline documentation.