Wine Store Project Documentation

Table of Contents

1. Project Setup and Commands

Initial Project Creation

```
# Create Next.js project with TypeScript
npx create-next-app@latest store-wine --typescript --tailwind --app
cd store-wine

# Install primary dependencies
npm install @prisma/client @radix-ui/react-checkbox @radix-ui/react-dropdown-
menu \
@radix-ui/react-icons @radix-ui/react-label @radix-ui/react-popover \
@radix-ui/react-select @radix-ui/react-separator @radix-ui/react-slot \
class-variance-authority clsx embla-carousel-react lucide-react next-themes \
react-icons sonner tailwind-merge

# Install development dependencies
npm install -D @eslint/eslintrc @tailwindcss/postcss @types/node @types/react \
@types/react-dom csv-parse eslint eslint-config-next prisma tailwindcss \
tw-animate-css typescript
```

Prisma Database Setup

```
# Initialize Prisma
npx prisma init

# After creating schema.prisma...
npx prisma generate

# Create initial migration
npx prisma migrate dev --name init

# Push schema to database
npx prisma db push

# Seed database
npx prisma db seed
```

Development Commands

```
# Start development server with Turbopack
npm run dev

# Build for production
npm run build

# Start production server
npm start

# Run linting
npm run lint

# Run database seeding
npm run seed
```

2. Project Overview

A full-stack e-commerce application for a wine store built with Next.js 14, featuring:

- Responsive product grid with hover effects
- Product favoriting system
- User authentication
- Product rating system
- Admin dashboard
- Shopping cart functionality
- Supabase backend integration
- Prisma ORM for database management

3. Tech Stack

Frontend

- Next.js 14 with App Router
- React 18
- TypeScript
- Tailwind CSS
- Shadcn UI Components
- React Icons

Backend

- Supabase (PostgreSQL)
- Prisma ORM
- Next.js API Routes

Development Tools

- ESLint
- PostCSS
- TypeScript
- Turbopack

4. Project Structure

```
store-wine/
├── .next/                      # Next.js build output
│   ├── cache/
│   ├── server/
│   │   └── app/
│   │       ├── _not-found/
│   │       ├── about/
│   │       ├── favicon.ico/
│   │       ├── favorites/
│   │       ├── page/
│   │       └── products/
│   ├── static/
│   │   ├── chunks/
│   │   ├── development/
│   │   └── media/
│   └── types/
├── app/                        # Next.js app router pages
│   ├── about/                  # About page
│   ├── admin/                  # Admin dashboard (planned)
│   ├── cart/                   # Shopping cart (planned)
│   ├── favorites/              # Favorites page
│   ├── orders/                 # Orders management (planned)
│   ├── products/               # Products listing
│   ├── reviews/                # Product reviews (planned)
│   ├── layout.tsx             # Root layout
│   ├── page.tsx               # Home page
│   ├── providers.tsx          # App providers
│   └── globals.css            # Global styles
├── components/                 # React components
│   ├── cart/                   # Cart components (planned)
│   ├── form/                   # Form components
│   ├── global/                 # Global components
│   │   ├── Container.tsx      # Layout container (341B, 23 lines)
│   │   ├── EmptyList.tsx      # Empty state (248B, 14 lines)
│   │   ├── SectionTitle.tsx   # Section headers (298B, 14 lines)
│   │   └── LoadingContainer.tsx # Loading state (132B, 8 lines)
│   ├── home/                   # Home page components
│   │   ├── FeaturedProducs.tsx # Featured products (532B, 16 lines)
│   │   ├── HeroCarousel.tsx   # Hero slider (120B, 8 lines)
│   │   └── Hero.tsx           # Hero section (96B, 8 lines)
│   ├── navbar/                 # Navigation components
│   │   ├── Logo.tsx           # Site logo (584B, 25 lines)
│   │   ├── NavSearch.tsx      # Search bar (253B, 13 lines)
│   │   ├── Navbar.tsx         # Main navigation (1.1KB, 38 lines)
│   │   ├── DarkMode.tsx       # Theme toggle (1.2KB, 41 lines)
│   │   ├── UserIcon.tsx       # User menu icon (108B, 8 lines)
│   │   ├── LinksDropdown.tsx  # Mobile menu (1003B, 36 lines)
│   │   ├── SignOutLink.tsx    # Logout button (117B, 8 lines)
│   │   └── CartButton.tsx     # Cart icon (635B, 24 lines)
│   ├── products/               # Product components
│   │   ├── ProductsGrid.tsx   # Product grid (2.3KB, 60 lines)
│   │   ├── FavoriteToggleButton.tsx # Favorite button (322B, 11 lines)
│   │   ├── ProdcutsList.tsx   # Product list view (120B, 8 lines)
│   │   ├── ProductsContainer.tsx # Products wrapper (135B, 8 lines)
│   │   └── FavoredToggleForm.tsx # Favorite toggle (135B, 8 lines)
│   ├── single-product/         # Product detail (planned)
│   └── ui/                     # Shadcn UI components
├── lib/                        # Library code
│   └── generated/              # Generated types
├── prisma/                     # Database configuration
│   ├── migrations/
│   │   └── 0_init/            # Initial migration
│   ├── schema.prisma          # Database schema
│   ├── seed.js                # Seed script
│   ├── seed-images.js         # Image seeding
│   ├── wine100.csv            # Wine data
│   └── rating1000.csv         # Ratings data
├── public/                     # Static assets
│   └── images/
│       └── wines/             # Product images
├── utils/                      # Utility functions
├── .env                        # Environment variables
├── .gitignore                 # Git ignore rules
├── components.json             # Shadcn UI configuration
├── eslint.config.mjs         # ESLint configuration
├── next.config.ts            # Next.js configuration
├── package.json              # Project dependencies
├── postcss.config.mjs        # PostCSS configuration
├── README.md                 # Project readme
└── tsconfig.json             # TypeScript configuration
```

5. Environment Configuration

Current Environment Variables (env file)

```
# Environment variables declared in this file are automatically made available
to Prisma.
# See the documentation for more detail: https://pris.ly/d/prisma-
schema#accessing-environment-variables-from-the-schema

# Prisma supports the native connection string format for PostgreSQL, MySQL,
SQLite, SQL Server, MongoDB and CockroachDB.
# See the documentation for all the connection string options:
https://pris.ly/d/connection-strings

DATABASE_URL="postgresql://postgres.kppejyvppvltypgrolmv:Bizerte5678924@aws-0-
us-east-2.pooler.supabase.com:6543/postgres?pgbouncer=true&connection_limit=1"
DIRECT_URL="postgresql://postgres.kppejyvppvltypgrolmv:Bizerte5678924@aws-0-us-
east-2.pooler.supabase.com:5432/postgres"
```

6. Database Schema and Commands

Prisma Schema

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider  = "postgresql"
  url       = env("DATABASE_URL")
  directUrl = env("DIRECT_URL")
}

model Wine {
  abv       Float
  acidity   String
  body      String
  code      String
  elaborate String?
  grapes    String
  harmonize String
  id        Int      @id @default(autoincrement())
  name      String
  price     Int      @default(0)
  regionId  Int
  type      String
  createdAt DateTime @default(now())
  featured  Boolean  @default(false)
  images    Image[]
  ratings   Rating[]
  region    Region   @relation("RegionToWine", fields: [regionId], references:
[id])
}

model Region {
  id      Int    @id @default(autoincrement())
  name    String
  country String
  wines   Wine[] @relation("RegionToWine")
}

model Rating {
  id      Int      @id @default(autoincrement())
  wineId  Int
  userId  Int
  vintage String?
  rating  Float
  date    DateTime
  user    User     @relation(fields: [userId], references: [id])
  wine    Wine     @relation(fields: [wineId], references: [id])
}

model User {
  email    String   @unique
  id       Int      @id @default(autoincrement())
  name     String?
  password String
  role     String   @default("USER")
  ratings  Rating[]
}

model Image {
  id     Int    @id @default(autoincrement())
  url    String
  wineId Int
  wine   Wine   @relation(fields: [wineId], references: [id])
}
```

Database Management Commands

```
# Access Prisma Studio
npx prisma studio

# Reset database
npx prisma migrate reset

# Pull database schema
npx prisma db pull

# Push schema changes
npx prisma db push

# Create new migration
npx prisma migrate dev --name <migration-name>

# Apply migrations (production)
npx prisma migrate deploy

# Database backup
pg_dump -h aws-0-us-east-2.pooler.supabase.com -U postgres.kppejyvppvltypgrolmv
-d postgres > backup.sql

# Database restore
psql -h aws-0-us-east-2.pooler.supabase.com -U postgres.kppejyvppvltypgrolmv -d
postgres < backup.sql
```

7. Features Implementation

Implemented Features

Product Grid (ProductsGrid.tsx)

- Responsive 4-column grid layout
- Dynamic image sizing
- Hover effects:
  - Image scale (1.3x)
  - Card shadow
- Favorite button overlay
- Two-line product name truncation
- Price display with formatting

Navigation (Navbar.tsx)

- Responsive layout
- Dark mode toggle
- Shopping cart indicator
- Search functionality
- Mobile dropdown menu

Database Integration

- Prisma ORM setup
- Supabase connection
- Initial migrations
- Seed data implementation

Planned Features

- Shopping cart functionality
- Checkout process
- User authentication
- Admin dashboard
- Order management
- Review system
- Single product view
- User favorites management

8. Development Guidelines

Code Organization

- Components organized by feature
- Shared components in `components/global`
- UI components in `components/ui`
- Utility functions in `utils`
- Database operations in `lib`

Styling Conventions

- Tailwind CSS for styling
- Custom components using shadcn/ui
- Responsive breakpoints:
  - md: 768px
  - lg: 1024px
  - xl: 1280px

Best Practices

- TypeScript for type safety
- Component-based architecture
- Server-side rendering where possible
- Image optimization with Next.js Image
- Environment variable management
- Database connection pooling

9. Deployment Process

Pre-deployment Checklist

1. Update environment variables
2. Generate production build
3. Run database migrations
4. Verify dependencies
5. Check TypeScript errors
6. Run linting checks

Deployment Commands

```
# Install dependencies
npm install

# Generate Prisma Client
npx prisma generate

# Build application
npm run build

# Apply database migrations
npx prisma migrate deploy

# Start production server
npm start
```

10. Backup Procedures

Code Backup

```
# Create timestamped backup
timestamp=$(date +"%Y%m%d_%H%M%S")
mkdir -p ~/project_backups/store-wine_$timestamp

# Backup code excluding node_modules and .next
rsync -av --exclude 'node_modules' --exclude '.next' ./
~/project_backups/store-wine_$timestamp/

# Backup environment variables
cp .env ~/project_backups/store-wine_$timestamp/.env.backup
```

Database Backup

```
# Create backup directory
mkdir -p ~/project_backups/store-wine_$timestamp/database

# Export database
```

```
pg_dump -h aws-0-us-east-2.pooler.supabase.com \
-U postgres.kppejyvppvltypgrolmv \
-d postgres > ~/project_backups/store-wine_$timestamp/database/backup.sql

# Backup Prisma schema and migrations
cp -r prisma ~/project_backups/store-wine_$timestamp/
```

Restore Procedures

```
# Restore code
cp -r ~/project_backups/store-wine_$timestamp/* ./

# Restore database
psql -h aws-0-us-east-2.pooler.supabase.com \
-U postgres.kppejyvppvltypgrolmv \
-d postgres < ~/project_backups/store-wine_$timestamp/database/backup.sql

# Install dependencies
npm install

# Generate Prisma client
npx prisma generate

# Apply migrations
npx prisma migrate deploy
```

```

```