# VisionGuard - Premium Blue Light Blocking Glasses Store

## Overview

VisionGuard is a full-featured e-commerce website specializing in premium blue light blocking glasses. Built with React, TypeScript, and powered by Supabase with integrated Stripe payment processing, this application provides a complete shopping experience for customers seeking to protect their eyes from digital eye strain.

🌐 **Live Demo**: [Visit VisionGuard Store](https://your-deployed-url.com)

## Features

### 🛍️ Complete E-Commerce Solution

* **Product Catalog**: Browse premium blue light blocking glasses collection
* **Advanced Filtering**: Filter by style, price, and features
* **Product Details**: Comprehensive product pages with specifications
* **Shopping Cart**: Persistent cart with add/remove functionality
* **Real Payment Processing**: Secure Stripe integration for credit card payments
* **Order Management**: Complete order tracking and confirmation system

### 🔒 Secure Payment System

* **Stripe Integration**: PCI-compliant payment processing
* **Payment Intents**: Secure payment confirmation flow
* **Order Confirmation**: Automated email confirmations
* **Real-time Status**: Payment and order status tracking
* **Refund Capability**: Full refund processing through admin panel

### 👥 User Experience

* **Responsive Design**: Optimized for all device sizes
* **Fast Loading**: Optimized performance with Vite
* **Modern UI**: Clean, intuitive interface with Tailwind CSS
* **Smooth Navigation**: Single-page application with React Router
* **Interactive Elements**: Dynamic cart updates and notifications

### 🔧 Admin Features

* **Product Management**: Add, edit, and remove products
* **Order Management**: View and manage customer orders
* **Payment Tracking**: Monitor payment statuses and transactions
* **Inventory Control**: Track product stock levels

## Technology Stack

### Frontend

* **React 18**: Modern React with hooks and functional components
* **TypeScript**: Type-safe development environment
* **Vite**: Lightning-fast build tool and development server
* **React Router DOM**: Client-side routing
* **Tailwind CSS**: Utility-first CSS framework
* **Lucide React**: Modern icon library
* **React Hot Toast**: Toast notifications

### Backend & Services

* **Supabase**: Backend-as-a-Service providing:
  + PostgreSQL database
  + Real-time subscriptions
  + Row Level Security (RLS)
  + Edge Functions for serverless APIs
* **Stripe**: Payment processing platform
* **Edge Functions**: Custom API endpoints for payment processing

### Payment Processing

* **@stripe/stripe-js**: Stripe JavaScript SDK
* **@stripe/react-stripe-js**: React components for Stripe Elements
* **Stripe Elements**: Secure card input components
* **Payment Intents API**: Modern payment confirmation flow

## Project Structure

visionguard/  
├── public/  
│ ├── images/ # Product images and assets  
│ └── videos/ # Background and promotional videos  
├── src/  
│ ├── components/ # Reusable React components  
│ │ ├── ui/ # UI components (buttons, cards, etc.)  
│ │ ├── Header.tsx # Navigation header  
│ │ ├── Footer.tsx # Footer component  
│ │ ├── ProductCard.tsx # Product display cards  
│ │ └── CheckoutForm.tsx # Stripe payment form  
│ ├── pages/ # Page components  
│ │ ├── HomePage.tsx # Main landing page  
│ │ ├── ProductPage.tsx # Individual product pages  
│ │ ├── CartPage.tsx # Shopping cart page  
│ │ ├── CheckoutPage.tsx # Checkout and payment  
│ │ ├── SuccessPage.tsx # Order confirmation  
│ │ └── AdminDashboard.tsx # Admin interface  
│ ├── contexts/ # React context providers  
│ │ └── CartContext.tsx # Cart state management  
│ ├── hooks/ # Custom React hooks  
│ ├── lib/ # Utilities and configurations  
│ │ ├── supabase.ts # Supabase client setup  
│ │ └── utils.ts # Helper functions  
│ └── App.tsx # Main application component  
├── supabase/ # Supabase configuration  
│ └── functions/ # Edge Functions  
│ ├── create-payment-intent/  
│ ├── confirm-payment/  
│ └── send-order-confirmation/  
├── package.json # Dependencies and scripts  
├── tailwind.config.js # Tailwind CSS configuration  
├── tsconfig.json # TypeScript configuration  
└── vite.config.ts # Vite configuration

## Getting Started

### Prerequisites

* Node.js (v18 or higher)
* pnpm, npm, or yarn
* Supabase account
* Stripe account

### Installation

1. **Clone the repository**

* git clone https://github.com/JFKOLLIE/postureguardpro.git  
  cd postureguardpro

1. **Install dependencies**

* pnpm install  
  # or  
  npm install

1. **Set up environment variables** Create a .env.local file in the root directory:

* VITE\_SUPABASE\_URL=your\_supabase\_project\_url  
  VITE\_SUPABASE\_ANON\_KEY=your\_supabase\_anon\_key  
  VITE\_STRIPE\_PUBLISHABLE\_KEY=your\_stripe\_publishable\_key

1. **Set up Supabase**
   * Create a new Supabase project
   * Run the database migrations (see Database Schema section)
   * Deploy the Edge Functions
   * Configure environment variables in Supabase dashboard:
   * STRIPE\_SECRET\_KEY=your\_stripe\_secret\_key  
     SUPABASE\_SERVICE\_ROLE\_KEY=your\_service\_role\_key  
     SUPABASE\_URL=your\_supabase\_url
2. **Start the development server**

* pnpm dev  
  # or  
  npm run dev

1. **Open your browser** Navigate to http://localhost:5173

### Building for Production

# Build the application  
pnpm build  
  
# Preview the production build  
pnpm preview

## Database Schema

### Products Table

CREATE TABLE products (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 name VARCHAR NOT NULL,  
 price DECIMAL(10,2) NOT NULL,  
 description TEXT,  
 image\_url VARCHAR,  
 features TEXT[],  
 in\_stock BOOLEAN DEFAULT true,  
 created\_at TIMESTAMP DEFAULT NOW(),  
 updated\_at TIMESTAMP DEFAULT NOW()  
);

### Orders Table

CREATE TABLE orders (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 user\_id UUID,  
 stripe\_payment\_intent\_id VARCHAR UNIQUE,  
 status VARCHAR DEFAULT 'pending',  
 total\_amount DECIMAL(10,2) NOT NULL,  
 currency VARCHAR DEFAULT 'usd',  
 customer\_email VARCHAR,  
 shipping\_address JSONB,  
 billing\_address JSONB,  
 created\_at TIMESTAMP DEFAULT NOW(),  
 updated\_at TIMESTAMP DEFAULT NOW()  
);

### Order Items Table

CREATE TABLE order\_items (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 order\_id UUID REFERENCES orders(id),  
 product\_id UUID REFERENCES products(id),  
 quantity INTEGER NOT NULL,  
 price\_at\_time DECIMAL(10,2) NOT NULL,  
 product\_name VARCHAR NOT NULL,  
 product\_image\_url VARCHAR,  
 created\_at TIMESTAMP DEFAULT NOW()  
);

## Stripe Integration

### Payment Flow

1. **Cart to Checkout**: User adds items to cart and proceeds to checkout
2. **Payment Intent**: Frontend calls Supabase Edge Function to create Stripe Payment Intent
3. **Order Creation**: Order and order items are created in database
4. **Payment Confirmation**: Stripe Elements handles secure card input
5. **Payment Processing**: Stripe processes the payment
6. **Order Confirmation**: Backend confirms payment and sends confirmation email

### Edge Functions

#### create-payment-intent

* Creates Stripe Payment Intent
* Calculates order total
* Creates order record in database
* Returns client secret for frontend

#### confirm-payment

* Verifies payment with Stripe
* Updates order status
* Handles payment confirmation

#### send-order-confirmation

* Sends order confirmation email
* Updates customer with order details

## Key Features

### Product Management

* Dynamic product loading from Supabase
* Image optimization and lazy loading
* Real-time stock tracking
* Product filtering and search

### Shopping Cart

* Persistent cart state using React Context
* Local storage backup
* Real-time total calculation
* Quantity management

### Payment Security

* PCI DSS compliant payment processing
* Secure card tokenization
* Payment Intent confirmation
* Fraud protection through Stripe Radar

### Order Management

* Real-time order status updates
* Email confirmations
* Order history tracking
* Admin order management dashboard

## Performance Optimizations

* **Code Splitting**: Dynamic imports for route-based code splitting
* **Image Optimization**: WebP format with fallbacks
* **Bundle Analysis**: Vite bundle analyzer for optimization
* **Lazy Loading**: Components and images loaded on demand
* **Caching**: Supabase query caching and browser cache optimization

## Security Features

* **Row Level Security**: Supabase RLS policies
* **Environment Variables**: Secure API key management
* **HTTPS**: SSL/TLS encryption
* **Input Validation**: Form and payment validation
* **CSRF Protection**: Built-in React CSRF protection

## Browser Support

* Chrome (latest 2 versions)
* Firefox (latest 2 versions)
* Safari (latest 2 versions)
* Edge (latest 2 versions)
* Mobile Safari (iOS 14+)
* Chrome Mobile (Android 8+)

## Contributing

1. Fork the repository
2. Create a feature branch (git checkout -b feature/amazing-feature)
3. Commit your changes (git commit -m 'Add amazing feature')
4. Push to the branch (git push origin feature/amazing-feature)
5. Open a Pull Request

### Development Guidelines

* Follow TypeScript best practices
* Use Tailwind CSS for styling
* Write comprehensive tests
* Follow React hooks best practices
* Ensure mobile responsiveness
* Test payment flows in Stripe test mode

## Deployment

### Frontend Deployment

The application can be deployed to: - Vercel - Netlify - GitHub Pages - Any static hosting service

### Backend Requirements

* Supabase project with Edge Functions enabled
* Stripe account with payment processing enabled
* Environment variables configured in hosting platform

## Environment Variables

### Frontend (.env.local)

VITE\_SUPABASE\_URL=your\_supabase\_url  
VITE\_SUPABASE\_ANON\_KEY=your\_supabase\_anon\_key  
VITE\_STRIPE\_PUBLISHABLE\_KEY=pk\_test\_...

### Supabase Edge Functions

STRIPE\_SECRET\_KEY=sk\_test\_...  
SUPABASE\_SERVICE\_ROLE\_KEY=your\_service\_role\_key  
SUPABASE\_URL=your\_supabase\_url

## Testing

### Payment Testing

Use Stripe test cards for development:

# Successful payment  
4242 4242 4242 4242  
  
# Declined payment  
4000 0000 0000 0002  
  
# 3D Secure required  
4000 0025 0000 3155

### Running Tests

# Run unit tests  
pnpm test  
  
# Run E2E tests  
pnpm test:e2e  
  
# Run payment flow tests  
pnpm test:payments

## License

This project is proprietary software. All rights reserved.

## Support

For support and questions: - **Developer**: JFKOLLIE - **Email**: jfkollie@gmail.com - **Repository**: [GitHub Issues](https://github.com/JFKOLLIE/postureguardpro/issues)

## Acknowledgments

* **React Team**: For the amazing React framework
* **Supabase Team**: For the excellent backend-as-a-service platform
* **Stripe Team**: For secure and reliable payment processing
* **Tailwind CSS**: For the utility-first CSS framework
* **Vite Team**: For the fast build tool and development experience

**VisionGuard - Protecting Your Eyes in the Digital Age** 👁️✨

Experience crystal-clear vision and reduced eye strain with our premium blue light blocking glasses, designed for the modern digital lifestyle.