LANGUAGECONTEXT.TSX:

'use client'

import { useState } from 'react'

import { useRouter } from 'next/navigation'

import { Globe } from 'lucide-react'

import {

DropdownMenu,

DropdownMenuContent,

DropdownMenuItem,

DropdownMenuTrigger,

} from "@/components/ui/dropdown-menu"

const languages = [

{ code: 'en', name: 'English' },

{ code: 'es', name: 'Español' },

{ code: 'fr', name: 'Français' },

{ code: 'de', name: 'Deutsch' },

]

export default function LanguageSelector() {

const router = useRouter()

const [currentLang, setCurrentLang] = useState('en')

const handleLanguageChange = (langCode: string) => {

setCurrentLang(langCode)

// Update the URL with the new language

router.push(/${langCode}${window.location.pathname})

}

return (

<DropdownMenu>

<DropdownMenuTrigger className="flex items-center gap-2">

<Globe className="w-5 h-5" />

<span>{languages.find(lang => lang.code === currentLang)?.name}</span>

</DropdownMenuTrigger>

<DropdownMenuContent>

{languages.map((lang) => (

<DropdownMenuItem

key={lang.code}

onClick={() => handleLanguageChange(lang.code)}

>

{lang.name}

</DropdownMenuItem>

))}

</DropdownMenuContent>

</DropdownMenu>

)

}

PRODUCTGRID:

'use client'

import Link from 'next/link'

import { useCart } from '../context/cartContext/page'

import { useWishlist } from '../context/wishlistContext/page'

import { Heart, ShoppingCart } from 'lucide-react'

interface Product {

id: number

name: string

price: number

stock: number

image: string

}

interface ProductGridProps {

products: Product[]

}

export default function ProductGrid({ products }: ProductGridProps) {

const { addToCart } = useCart()

const { addToWishlist, removeFromWishlist, isInWishlist } = useWishlist()

return (

<div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6">

{products.map((product) => (

<div key={product.id} className="border rounded-lg overflow-hidden shadow-lg">

<Link href={/product/${product.id}}>

<img src={product.image || "/placeholder.svg"} alt={product.name} className="w-full h-48 object-cover" />

</Link>

<div className="p-4">

<Link href={/product/${product.id}}>

<h2 className="text-lg font-semibold mb-2">{product.name}</h2>

</Link>

<p className="text-gray-600 mb-2">${product.price.toFixed(2)}</p>

<p className="text-sm text-gray-500 mb-4">

{product.stock > 0 ? ${product.stock} in stock : 'Out of stock'}

</p>

<div className="flex justify-between items-center">

<button

onClick={() => addToCart({

id: product.id, name: product.name, price: product.price, quantity: 1,

image: ''

})}

disabled={product.stock === 0}

className="bg-blue-500 text-white px-4 py-2 rounded-full hover:bg-blue-600 transition duration-300 disabled:bg-gray-400"

>

<ShoppingCart className="inline-block mr-2" size={16} />

Add to Cart

</button>

<button

onClick={() => isInWishlist(product.id) ? removeFromWishlist(product.id) : addToWishlist(product)}

className={p-2 rounded-full ${isInWishlist(product.id) ? 'text-red-500' : 'text-gray-500'} hover:bg-gray-100 transition duration-300}

>

<Heart className="inline-block" size={20} />

</button>

</div>

</div>

</div>

))}

</div>

)

}

NAVBAR:

'use client'

import Link from 'next/link'

import { useState } from 'react'

import { useLanguage } from '../context/LanguageContext'

import { useUser } from '../context/UserContext'

import { useCart } from '../context/cartContext/page'

import { useWishlist } from '../context/wishlistContext/page'

import { Search, ShoppingCart, Heart, User } from 'lucide-react'

export default function Header() {

const [searchQuery, setSearchQuery] = useState('')

const { language, setLanguage } = useLanguage()

const { user } = useUser()

const { cart } = useCart()

const { wishlist } = useWishlist()

const handleSearch = (e: React.FormEvent) => {

e.preventDefault()

// Implement search functionality

console.log('Searching for:', searchQuery)

}

return (

<header className="bg-white shadow-md">

<div className="container mx-auto px-4 py-4">

<div className="flex items-center justify-between">

<Link href="/" className="text-2xl font-bold">E-commerce Platform</Link>

<form onSubmit={handleSearch} className="flex-1 mx-4">

<div className="relative">

<input

type="text"

placeholder="Search products..."

value={searchQuery}

onChange={(e) => setSearchQuery(e.target.value)}

className="w-full p-2 pl-10 rounded-full border border-gray-300 focus:outline-none focus:ring-2 focus:ring-blue-500"

/>

<Search className="absolute left-3 top-1/2 transform -translate-y-1/2 text-gray-400" />

</div>

</form>

<nav className="flex items-center space-x-4">

<Link href="/cart" className="relative">

<ShoppingCart />

{cart.length > 0 && (

<span className="absolute -top-2 -right-2 bg-red-500 text-white rounded-full w-5 h-5 flex items-center justify-center text-xs">

{cart.length}

</span>

)}

</Link>

<Link href="/wishlist" className="relative">

<Heart />

{wishlist.length > 0 && (

<span className="absolute -top-2 -right-2 bg-red-500 text-white rounded-full w-5 h-5 flex items-center justify-center text-xs">

{wishlist.length}

</span>

)}

</Link>

<Link href={user ? "/profile" : "/login"}>

<User />

</Link>

<select

value={language}

onChange={(e) => setLanguage(e.target.value)}

className="p-2 rounded border border-gray-300 focus:outline-none focus:ring-2 focus:ring-blue-500"

>

<option value="en">English</option>

<option value="es">Español</option>

<option value="fr">Français</option>

</select>

</nav>

</div>

</div>

</header>

)

}

CHECKOUT:

'use client'

import { useState } from 'react'

import { useCart } from '@/app/context/cartContext/page'

import Button from "@/app/components/button"

import { Input } from "@/app/components/input"

import { Label } from 'recharts'

const shippingOptions = [

{ id: 'standard', name: 'Standard Shipping', price: 5.99 },

{ id: 'express', name: 'Express Shipping', price: 14.99 },

]

export default function CheckoutPage() {

const { cart, getCartTotal, clearCart } = useCart()

const [formData, setFormData] = useState({

name: '',

email: '',

address: '',

city: '',

country: '',

zipCode: '',

})

const [selectedShipping, setSelectedShipping] = useState(shippingOptions[0].id)

const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {

setFormData({ ...formData, [e.target.name]: e.target.value })

}

const handleSubmit = async (e: React.FormEvent) => {

e.preventDefault()

try {

const response = await fetch('/api/create-checkout-session', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({

items: cart,

shipping: shippingOptions.find(option => option.id === selectedShipping),

customerInfo: formData,

}),

})

if (response.ok) {

const { sessionId } = await response.json()

// Redirect to Stripe Checkout

const stripe = await loadStripe(process.env.NEXT\_PUBLIC\_STRIPE\_PUBLISHABLE\_KEY!)

await stripe?.redirectToCheckout({ sessionId })

} else {

throw new Error('Failed to create checkout session')

}

} catch (error) {

console.error('Checkout error:', error)

alert('An error occurred during checkout')

}

}

const totalWithShipping = getCartTotal() + shippingOptions.find(option => option.id === selectedShipping)!.price

return (

<div className="container mx-auto px-4 py-12">

<h1 className="text-3xl font-bold mb-8">Checkout</h1>

<div className="grid md:grid-cols-2 gap-8">

<form onSubmit={handleSubmit} className="space-y-4">

<div>

<Label htmlFor="name">Full Name</Label>

<Input type="text" id="name" name="name" value={formData.name} onChange={handleChange} required />

</div>

<div>

<Label htmlFor="email">Email</Label>

<Input type="email" id="email" name="email" value={formData.email} onChange={handleChange} required />

</div>

<div>

<Label htmlFor="address">Address</Label>

<Input type="text" id="address" name="address" value={formData.address} onChange={handleChange} required />

</div>

<div>

<Label htmlFor="city">City</Label>

<Input type="text" id="city" name="city" value={formData.city} onChange={handleChange} required />

</div>

<div>

<Label htmlFor="country">Country</Label>

<Input type="text" id="country" name="country" value={formData.country} onChange={handleChange} required />

</div>

<div>

<Label htmlFor="zipCode">Zip Code</Label>

<Input type="text" id="zipCode" name="zipCode" value={formData.zipCode} onChange={handleChange} required />

</div>

<div>

<Label>Shipping Method</Label>

<RadioGroup value={selectedShipping} onValueChange={setSelectedShipping}>

{shippingOptions.map((option) => (

<div key={option.id} className="flex items-center space-x-2">

<RadioGroupItem value={option.id} id={option.id} />

<Label htmlFor={option.id}>{option.name} - ${option.price.toFixed(2)}</Label>

</div>

))}

</RadioGroup>

</div>

<Button type="submit" className="w-full mt-8">Place Order</Button>

</form>

<div>

<h2 className="text-2xl font-bold mb-4">Order Summary</h2>

<div className="space-y-2">

{cart.map((item) => (

<div key={item.id} className="flex justify-between">

<span>{item.name} x {item.quantity}</span>

<span>${(item.price \* item.quantity).toFixed(2)}</span>

</div>

))}

<div className="border-t pt-2">

<div className="flex justify-between font-bold">

<span>Subtotal</span>

<span>${getCartTotal().toFixed(2)}</span>

</div>

<div className="flex justify-between">

<span>Shipping</span>

<span>${shippingOptions.find(option => option.id === selectedShipping)!.price.toFixed(2)}</span>

</div>

<div className="flex justify-between font-bold text-lg mt-2">

<span>Total</span>

<span>${totalWithShipping.toFixed(2)}</span>

</div>

</div>

</div>

</div>

</div>

</div>

)

}

function loadStripe(arg0: string) {

throw new Error('Function not implemented.')

}

SIGN-UP:

'use client'

import { useState } from 'react'

import Button from "@/app/components/button"

import { Input } from "@/app/components/input"

import { Label } from 'recharts'

export default function SignUp() {

const [formData, setFormData] = useState({

name: '',

email: '',

password: '',

confirmPassword: '',

})

const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {

setFormData({ ...formData, [e.target.name]: e.target.value })

}

const handleSubmit = async (e: React.FormEvent) => {

e.preventDefault()

if (formData.password !== formData.confirmPassword) {

alert("Passwords don't match")

return

}

try {

const response = await fetch('/api/signup', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify(formData),

})

if (response.ok) {

alert('Sign up successful!')

// Redirect to login page or home page

} else {

const error = await response.text()

alert(Sign up failed: ${error})

}

} catch (error) {

console.error('Sign up error:', error)

alert('An error occurred during sign up')

}

}

return (

<div className="max-w-md mx-auto mt-8">

<h2 className="text-2xl font-bold mb-4">Sign Up</h2>

<form onSubmit={handleSubmit} className="space-y-4">

<div>

<Label htmlFor="name">Name</Label>

<Input type="text" id="name" name="name" value={formData.name} onChange={handleChange} required />

</div>

<div>

<Label htmlFor="email">Email</Label>

<Input type="email" id="email" name="email" value={formData.email} onChange={handleChange} required />

</div>

<div>

<Label htmlFor="password">Password</Label>

<Input type="password" id="password" name="password" value={formData.password} onChange={handleChange} required />

</div>

<div>

<Label htmlFor="confirmPassword">Confirm Password</Label>

<Input type="password" id="confirmPassword" name="confirmPassword" value={formData.confirmPassword} onChange={handleChange} required />

</div>

<Button type="submit" className="w-full">Sign Up</Button>

</form>

</div>

)

}

CART:

'use client'

import { useCart } from '@/app/context/cartContext/page'

import Button from "@/app/components/button"

import Link from 'next/link'

export default function CartPage() {

const { cart, removeFromCart, clearCart, getCartTotal } = useCart()

return (

<div className="container mx-auto px-4 py-12">

<h1 className="text-3xl font-bold mb-8">Your Cart</h1>

{cart.length === 0 ? (

<p>Your cart is empty.</p>

) : (

<>

<div className="space-y-4">

{cart.map((item) => (

<div key={item.id} className="flex items-center space-x-4 border-b pb-4">

<img src={item.image || "/placeholder.svg"} alt={item.name} className="w-16 h-16 object-cover rounded" />

<div className="flex-grow">

<h2 className="text-xl font-semibold">{item.name}</h2>

<p>Quantity: {item.quantity}</p>

<p>Price: ${item.price.toFixed(2)}</p>

</div>

<Button variant="destructive" onClick={() => removeFromCart(item.id)}>Remove</Button>

</div>

))}

</div>

<div className="mt-8">

<p className="text-xl font-bold">Total: ${getCartTotal().toFixed(2)}</p>

<div className="mt-4 space-x-4">

<Button onClick={clearCart} variant="outline">Clear Cart</Button>

<Link href="/checkout">

<Button>Proceed to Checkout</Button>

</Link>

</div>

</div>

</>

)}

</div>

)

}

CATEGORIESROW:

import Link from 'next/link'

const categories = [

{ name: 'Women', image: '/category-women.jpg' },

{ name: 'Men', image: '/category-men.jpg' },

{ name: 'Bags', image: '/category-bags.jpg' },

{ name: 'Accessories', image: '/category-accessories.jpg' },

{ name: 'Shoes', image: '/category-shoes.jpg' },

]

export default function CategoriesRow() {

return (

<div className="container mx-auto px-4 py-12">

<h2 className="text-3xl font-bold mb-8 text-center">Shop by Category</h2>

<div className="grid grid-cols-2 md:grid-cols-3 lg:grid-cols-5 gap-8">

{categories.map((category) => (

<Link href={/category/${category.name.toLowerCase()}} key={category.name} className="relative group">

<div className="aspect-square overflow-hidden rounded-full border-4 border-white shadow-lg">

<img

src={category.image || "/placeholder.svg"}

alt={category.name}

className="w-full h-full object-cover transition-transform duration-300 group-hover:scale-110"

/>

<div className="absolute inset-0 bg-black bg-opacity-40 flex items-center justify-center transition-opacity duration-300 opacity-0 group-hover:opacity-100 rounded-full">

<h3 className="text-white text-xl font-bold">{category.name}</h3>

</div>

</div>

</Link>

))}

</div>

</div>

)

}

FEATUREDPRODUCTS:

import Link from 'next/link'

import Button from "@/app/components/button"

const featuredProducts = [

{ id: 1, name: 'Summer Dress', description: 'Perfect for sunny days', price: 79.99, image: '/featured-product-1.jpg' },

{ id: 2, name: 'Leather Jacket', description: 'Timeless style', price: 199.99, image: '/featured-product-2.jpg' },

{ id: 3, name: 'Designer Sunglasses', description: 'Protect your eyes in style', price: 129.99, image: '/featured-product-3.jpg' },

{ id: 4, name: 'Classic Watch', description: 'Elegant timepiece', price: 149.99, image: '/featured-product-4.jpg' },

]

export default function FeaturedProducts() {

return (

<div className="container mx-auto px-4 py-12">

<h2 className="text-3xl font-bold mb-8 text-center">Featured Products</h2>

<div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-8">

{featuredProducts.map((product) => (

<div key={product.id} className="border rounded-lg overflow-hidden shadow-md">

<img src={product.image || "/placeholder.svg"} alt={product.name} className="w-full h-48 object-cover" />

<div className="p-4">

<h3 className="text-lg font-semibold mb-2">{product.name}</h3>

<p className="text-gray-600 mb-4">{product.description}</p>

<div className="flex justify-between items-center">

<span className="text-lg font-bold">${product.price.toFixed(2)}</span>

<Link href={/product/${product.id}}>

<Button>View Details</Button>

</Link>

</div>

</div>

</div>

))}

</div>

</div>

)

}

WISHLISTCONTEXT:

'use client'

import React, { createContext, useContext, useState, useEffect } from 'react'

interface WishlistItem {

id: number

name: string

price: number

image: string

}

interface WishlistContextType {

wishlist: WishlistItem[]

addToWishlist: (item: WishlistItem) => void

removeFromWishlist: (id: number) => void

isInWishlist: (id: number) => boolean

}

const WishlistContext = createContext<WishlistContextType | undefined>(undefined)

export function WishlistProvider({ children }: { children: React.ReactNode }) {

const [wishlist, setWishlist] = useState<WishlistItem[]>([])

useEffect(() => {

const savedWishlist = localStorage.getItem('wishlist')

if (savedWishlist) {

setWishlist(JSON.parse(savedWishlist))

}

}, [])

useEffect(() => {

localStorage.setItem('wishlist', JSON.stringify(wishlist))

}, [wishlist])

const addToWishlist = (item: WishlistItem) => {

setWishlist(prevWishlist => {

if (!prevWishlist.some(wishlistItem => wishlistItem.id === item.id)) {

return [...prevWishlist, item]

}

return prevWishlist

})

}

const removeFromWishlist = (id: number) => {

setWishlist(prevWishlist => prevWishlist.filter(item => item.id !== id))

}

const isInWishlist = (id: number) => {

return wishlist.some(item => item.id === id)

}

return (

<WishlistContext.Provider value={{ wishlist, addToWishlist, removeFromWishlist, isInWishlist }}>

{children}

</WishlistContext.Provider>

)

}

export function useWishlist() {

const context = useContext(WishlistContext)

if (context === undefined) {

throw new Error('useWishlist must be used within a WishlistProvider')

}

return context

}

CARTCONTEXT:

'use client'

import React, { createContext, useContext, useState, useEffect } from 'react'

interface CartItem {

image: string

id: number

name: string

price: number

quantity: number

}

interface CartContextType {

cart: CartItem[]

addToCart: (item: CartItem) => void

removeFromCart: (id: number) => void

updateQuantity: (id: number, quantity: number) => void

clearCart: () => void

getCartTotal: () => number

}

const CartContext = createContext<CartContextType | undefined>(undefined)

export function CartProvider({ children }: { children: React.ReactNode }) {

const [cart, setCart] = useState<CartItem[]>([])

useEffect(() => {

const savedCart = localStorage.getItem('cart')

if (savedCart) {

setCart(JSON.parse(savedCart))

}

}, [])

useEffect(() => {

localStorage.setItem('cart', JSON.stringify(cart))

}, [cart])

const addToCart = (item: CartItem) => {

setCart(prevCart => {

const existingItem = prevCart.find(cartItem => cartItem.id === item.id)

if (existingItem) {

return prevCart.map(cartItem =>

cartItem.id === item.id

? { ...cartItem, quantity: cartItem.quantity + 1 }

: cartItem

)

}

return [...prevCart, { ...item, quantity: 1 }]

})

}

const removeFromCart = (id: number) => {

setCart(prevCart => prevCart.filter(item => item.id !== id))

}

const updateQuantity = (id: number, quantity: number) => {

setCart(prevCart =>

prevCart.map(item =>

item.id === id ? { ...item, quantity: quantity } : item

)

)

}

const clearCart = () => {

setCart([])

}

const getCartTotal = () => {

return cart.reduce((total, item) => total + item.price \* item.quantity, 0)

}

return (

<CartContext.Provider value={{ cart, addToCart, removeFromCart, updateQuantity, clearCart, getCartTotal }}>

{children}

</CartContext.Provider>

)

}

export function useCart() {

const context = useContext(CartContext)

if (context === undefined) {

throw new Error('useCart must be used within a CartProvider')

}

return context

}

USERCONTEXT:

'use client'

import React, { createContext, useContext, useState, useEffect } from 'react'

interface User {

id: number

name: string

email: string

}

interface UserContextType {

user: User | null

login: (user: User) => void

logout: () => void

}

const UserContext = createContext<UserContextType | undefined>(undefined)

export function UserProvider({ children }: { children: React.ReactNode }) {

const [user, setUser] = useState<User | null>(null)

useEffect(() => {

// Check for saved user data in localStorage on initial load

const savedUser = localStorage.getItem('user')

if (savedUser) {

setUser(JSON.parse(savedUser))

}

}, [])

const login = (userData: User) => {

setUser(userData)

localStorage.setItem('user', JSON.stringify(userData))

}

const logout = () => {

setUser(null)

localStorage.removeItem('user')

}

return (

<UserContext.Provider value={{ user, login, logout }}>

{children}

</UserContext.Provider>

)

}

export function useUser() {

const context = useContext(UserContext)

if (context === undefined) {

throw new Error('useUser must be used within a UserProvider')

}

return context

}

HELP.TSX:

'use client'

import { useState } from 'react'

import { Search } from 'lucide-react'

const faqs = [

{

question: "How do I place an order?",

answer: "To place an order, simply browse our products, add items to your cart, and proceed to checkout. Follow the steps to enter your shipping and payment information."

},

{

question: "What payment methods do you accept?",

answer: "We accept major credit cards (Visa, MasterCard, American Express), PayPal, and Apple Pay."

},

{

question: "How can I track my order?",

answer: "Once your order is shipped, you'll receive a confirmation email with a tracking number. You can use this number to track your package on our website or the carrier's website."

},

// Add more FAQs as needed

]

export default function HelpPage() {

const [searchQuery, setSearchQuery] = useState('')

const [filteredFaqs, setFilteredFaqs] = useState(faqs)

const handleSearch = (e: React.FormEvent) => {

e.preventDefault()

const filtered = faqs.filter(faq =>

faq.question.toLowerCase().includes(searchQuery.toLowerCase()) ||

faq.answer.toLowerCase().includes(searchQuery.toLowerCase())

)

setFilteredFaqs(filtered)

}

return (

<div>

<h1 className="text-3xl font-bold mb-6">Help & FAQ</h1>

<form onSubmit={handleSearch} className="mb-8">

<div className="relative">

<input

type="text"

placeholder="Search FAQs..."

value={searchQuery}

onChange={(e) => setSearchQuery(e.target.value)}

className="w-full p-2 pl-10 rounded-full border border-gray-300 focus:outline-none focus:ring-2 focus:ring-blue-500"

/>

<Search className="absolute left-3 top-1/2 transform -translate-y-1/2 text-gray-400" />

</div>

</form>

<div className="space-y-6">

{filteredFaqs.map((faq, index) => (

<div key={index} className="bg-white shadow rounded-lg p-6">

<h2 className="text-xl font-semibold mb-2">{faq.question}</h2>

<p className="text-gray-600">{faq.answer}</p>

</div>

))}

</div>

</div>

)

}

PROFILE.TSX:

'use client'

import { useUser } from '../context/UserContext'

import { useState, useEffect } from 'react'

interface Order {

id: number

date: string

total: number

status: string

}

export default function UserProfile() {

const { user } = useUser()

const [orders, setOrders] = useState<Order[]>([])

useEffect(() => {

// Fetch user's orders from API

const fetchOrders = async () => {

const res = await fetch('/api/orders')

const data = await res.json()

setOrders(data)

}

fetchOrders()

}, [])

if (!user) {

return <div>Please log in to view your profile.</div>

}

return (

<div>

<h1 className="text-3xl font-bold mb-6">User Profile</h1>

<div className="bg-white shadow rounded-lg p-6 mb-6">

<h2 className="text-xl font-semibold mb-4">Account Information</h2>

<p><strong>Name:</strong> {user.name}</p>

<p><strong>Email:</strong> {user.email}</p>

</div>

<div className="bg-white shadow rounded-lg p-6">

<h2 className="text-xl font-semibold mb-4">Order History</h2>

{orders.length === 0 ? (

<p>No orders found.</p>

) : (

<table className="w-full">

<thead>

<tr>

<th className="text-left">Order ID</th>

<th className="text-left">Date</th>

<th className="text-left">Total</th>

<th className="text-left">Status</th>

</tr>

</thead>

<tbody>

{orders.map((order) => (

<tr key={order.id}>

<td>{order.id}</td>

<td>{order.date}</td>

<td>${order.total.toFixed(2)}</td>

<td>{order.status}</td>

</tr>

))}

</tbody>

</table>

)}

</div>

</div>

)

}

PRODUCTNAVBAR ;:

import Link from 'next/link'

const navItems = [

{ name: 'Accessories', href: '/accessories' },

{ name: 'Bags', href: '/bags' },

{ name: 'Shoes', href: '/shoes' },

{ name: 'New-in', href: '/new-in' },

]

export default function ProductNavbar() {

return (

<nav className="bg-gray-100 shadow-md">

<div className="container mx-auto px-4">

<ul className="flex justify-center space-x-8">

{navItems.map((item) => (

<li key={item.name}>

<Link href={item.href} className="block py-4 text-gray-700 hover:text-black font-medium">

{item.name}

</Link>

</li>

))}

</ul>

</div>

</nav>

)

}

CHECKOUTAPI:

import { NextResponse } from 'next/server'

import Stripe from 'stripe'

const stripe = new Stripe(process.env.STRIPE\_SECRET\_KEY!, {

apiVersion: '2022-11-15',

})

export async function POST(req: Request) {

try {

const { items, shipping, customerInfo } = await req.json()

const session = await stripe.checkout.sessions.create({

payment\_method\_types: ['card'],

line\_items: items.map((item: any) => ({

price\_data: {

currency: 'usd',

product\_data: {

name: item.name,

},

unit\_amount: Math.round(item.price \* 100),

},

quantity: item.quantity,

})),

mode: 'payment',

success\_url: ${process.env.NEXT\_PUBLIC\_BASE\_URL}/order-confirmation?session\_id={CHECKOUT\_SESSION\_ID},

cancel\_url: ${process.env.NEXT\_PUBLIC\_BASE\_URL}/cart,

shipping\_options: [

{

shipping\_rate\_data: {

type: 'fixed\_amount',

fixed\_amount: {

amount: Math.round(shipping.price \* 100),

currency: 'usd',

},

display\_name: shipping.name,

},

},

],

shipping\_address\_collection: {

allowed\_countries: ['US', 'CA'],

},

metadata: {

...customerInfo,

},

})

return NextResponse.json({ sessionId: session.id })

} catch (error) {

console.error('Checkout session error:', error)

return NextResponse.json({ message: 'An error occurred during checkout' }, { status: 500 })

}

}