

Marketplace E-Commerce

Hekta Furniture Hackathon
- Day 4

Template-4

Building Dynamic Frontend Components

Prepared by Owais Qazi

E-Commerce Marketplace - Hekta Furniture (Template-4)

Hackathon Day 4 Submission Report

1. Introduction

Project Name: E-Commerce Marketplace - Hekta Furniture (Template-4)

Tech Stack: Next.js 14.2.2, Sanity CMS, Vercel, GitHub, Tailwind CSS, ShadCN UI, Stripe Payment, Clerk Auth, Toast Notification, Sanity Color Plugin



Purpose: This project is an e-commerce marketplace template designed to showcase and sell furniture products, offering dynamic data fetching, filters, pagination, and an integrated payment system.

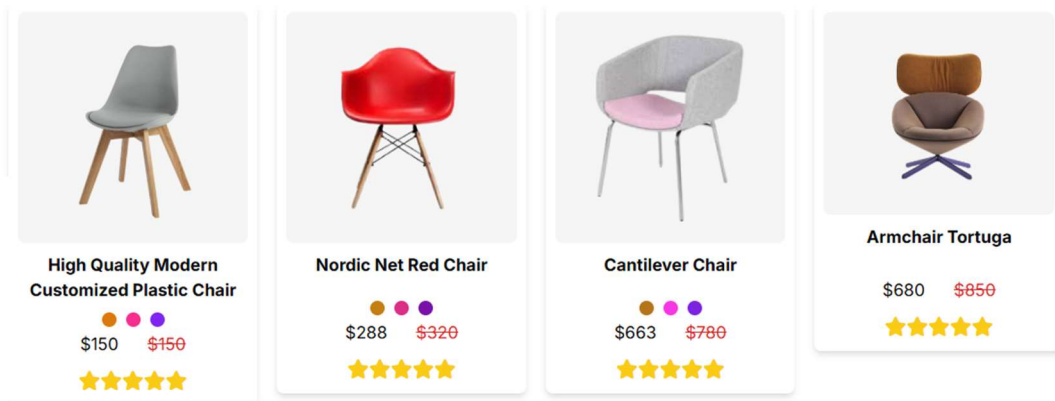
2. Functional Deliverables

2.1 Product Listing Page

- Dynamically fetches product data from Sanity CMS.
- Displays product images, names, prices, and ratings.

```
useEffect(() => {  
  const fetchProducts = async () => {  
    const query = `*[_type == "products"] {  
      name,  
      image{ asset->{ url } },  
      href,  
      price,  
      discountPercentage,  
      description,  
      colors,  
      rating_5,  
      rating_4,  
      rating_3,  
      rating_2,  
      rating_1,  
      createdAt  
    }`;  
  };  
  
  const result = await client.fetch<Item[]>(query);
```

n item Per Page: 1 Sort By: Best Match View:   Search...



2.2 Individual Product Detail Page

- Routes dynamically to each product using Next.js dynamic routing.
- Fetches and displays product details, including name, price, description, and available colors.

```
/* Shop Grid */  
<div className="lg:w-3/4 w-full grid grid-cols-1 gap-4 md:grid-cols-3 lg:grid-cols-4">  
  {getPaginatedProducts(currentPage).map((item) => (  
    <Link href={`~/products/${item._id}`} key={item._id}>  
      <div className="col-span-1">  
        <div className="flex flex-col h-full shadow-md p-2 rounded-md">  
          /* Image Section */  
          <div className="bg-stone-100 p-6 lg:p-4 md:p-6 rounded-md flex i
```

```

// Generate static params
export async function generateStaticParams() {
  const query = `*[_type == "products"]{ _id }`;
  const products = await client.fetch(query);

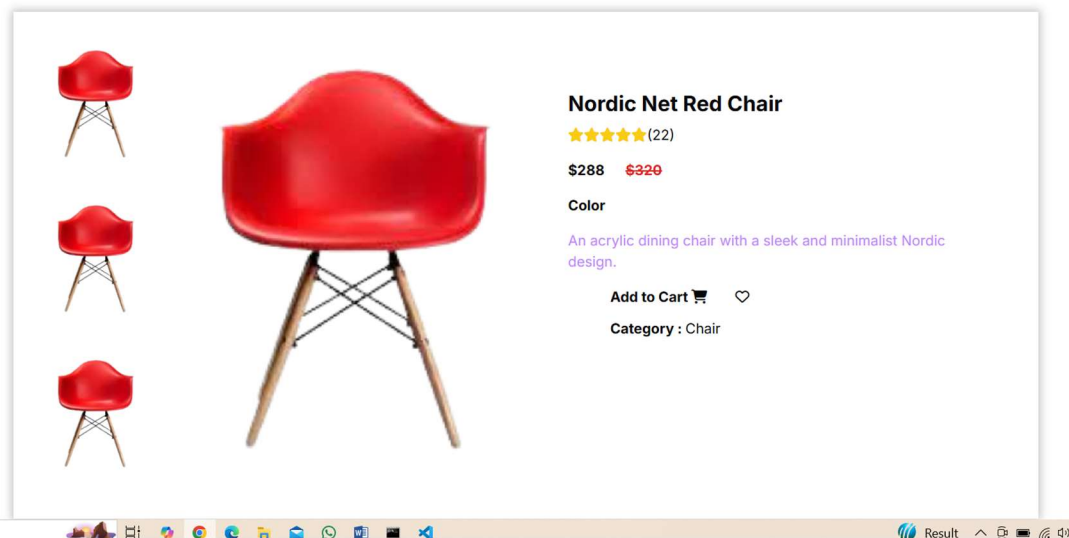
  return products.map((product: { _id: string }) => ({
    product: product._id,
  }));
}

// Fetch product data
async function getProduct(productId: string): Promise<ProductDet> {
  'use server';

  const query = `*[_type == "products" && _id == $id][0] {
    _id,
    name,
    description,
    image{ asset->{ url } },
    href,
    price,
    discountPercentage,
    colors,
    rating_5,
    rating_4,
    rating_3,
    rating_2,
    rating_1,
    createdOn,
    category
  }`;

  return client.fetch(query, { id: productId });
}

```



2.3 Filters, Search Bar, and Pagination

- Users can filter products by category, brand, discount offers, and ratings.
- Includes a fully functional search bar.
- Implements pagination to manage large datasets efficiently.

```
const filteredProducts = products.filter((product) => {
  const matchesSearch = product.name.toLowerCase().includes(searchQuery.toLowerCase());
  const matchesBrand = filters.brands.length === 0 || filters.brands.includes(product.brand);
  const matchesCategory =
    filters.categories.length === 0 || filters.categories.includes(product.category);
  const matchesDiscount =
    filters.discountOffers.length === 0 ||
    filters.discountOffers.some((discount) => product.discountPercentage >= parseInt(discount));
  const matchesRating =
    filters.ratings.length === 0 ||
    filters.ratings.some((rating) => calculateAverageRating(product) >= rating);

  return matchesSearch && matchesBrand && matchesCategory && matchesDiscount && matchesRating;
});
```

```
const ShopSidebar = ({ filters, setFilters }: any) => {

  const handleFilterChange = (type: any, value: any) => {
    setFilters((prev: any) => {
      const updatedFilters = { ...prev };
      if (updatedFilters[type].includes(value)) {
        updatedFilters[type] = updatedFilters[type].filter((item: any) => item !== value);
      } else {
        updatedFilters[type] = [...updatedFilters[type], value];
      }
      return updatedFilters;
    });
  };

  const [isOpen, setIsOpen] = useState(false);

  const toggleSidebar = () => {
    setIsOpen(!isOpen);
  };

  return (
    <div className="relative">
      {/* Filter Button positioned outside image container, centered in mobile view */}
      {isOpen && (
```

```

export default function ShopList({ searchQuery, sorting }: { searchQuery: string, sorting: string }) {
  useEffect(() => {
    const fetchProducts = async () => {
      const result = await client.fetch<Item[]>(query);
      // Sort by highest average rating first
      if(sorting === "bm"){
        result.sort((a, b) => calculateAverageRating(b) - calculateAverageRating(a));
      }
      if(sorting === "pl"){
        result.sort((a, b) => a.price - b.price);
      }
      if(sorting === "ph"){
        result.sort((a, b) => b.price - a.price);
      }
    }

    let filteredProducts = result;
    if (searchQuery) {
      filteredProducts = result.filter((product) =>
        product.name.toLowerCase().includes(searchQuery.toLowerCase())
      );
    }

    // Brand filter
    if (filters.brands.length > 0) {
      filteredProducts = filteredProducts.filter((product) =>
        filters.brands.includes(product.name)
      );
    }

    // Discount filter
    if (filters.discountOffers.length > 0) {
      filteredProducts = filteredProducts.filter((product) =>
        filters.discountOffers.some((offer) => {
          if (offer === "20% Cashback") return product.discountPercentage >= 20;
          if (offer === "5% Cashback Offer") return product.discountPercentage >= 5;
          if (offer === "25% Discount Offer") return product.discountPercentage >= 25;
          return false;
        })
      );
    }
  }, [searchQuery, filters, sorting]);
}

```

```

    // Ratings filter
    if (filters.ratings.length > 0) {
      filteredProducts = filteredProducts.filter((product) =>
        filters.ratings.some((rating) => calculateAverageRating(product) >= rating)
      );
    }

    const itemsPerPage = 8;
    const totalItems = filteredProducts.length;
    const pageCount = Math.ceil(totalItems / itemsPerPage);
    setTotalPages(pageCount);

    const startIndex = (currentPage - 1) * itemsPerPage;
    const paginatedProducts = filteredProducts.slice(startIndex, startIndex + itemsPerPage);

    setProducts(paginatedProducts);
  };
  fetchProducts();
}, [currentPage, searchQuery, filters, sorting]);

```

Product Brand


- ☐ Coaster Furniture
- ☐ Fusion Dot High Fashion
- ☐ Unique Furniture Restor
- ☐ Dream Furniture Flipping
- ☐ Young Repurposed
- ☐ Green DIY furniture

Discount Offer

- ☐ 20% Cashback
- ☐ 5% Cashback Offer
- ☐ 25% Discount Offer

Rating Item


- ☐ 5 stars
- ☐ 4 stars
- ☐ 3 stars
- ☐ 2 stars
- ☐ -



High Quality Modern Customized Plastic Chair

\$150 \$150


★★★★★



Nordic Net Red Chair

\$288 \$320


★★★★★



Cantilever Chair

\$663 \$780

★★★★★



Armchair Tortuga

\$680 \$850

★★★★★



Futuristic Sleek Modern Chair



Uchiwa Quilted Lounge






Rapson Thirty-Nine

Ecommerce Accessories & Fashion item
About 9,620 results (0.62 seconds)

Per Page: 1

Sort By: Best Match


View: ☐ ☐ ☐ Search...



High Quality Modern Customized Plastic Chair

\$150 \$150


★★★★★



Nordic Net Red Chair

\$288 \$320

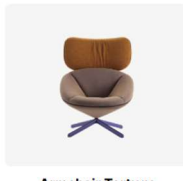
★★★★★



Cantilever Chair

\$663 \$780

★★★★★



Armchair Tortuga

\$680 \$850

★★★★★

Product Brand

- ☐ Coaster Furniture
- ☐ Fusion Dot High Fashion
- ☐ Unique Furniture Restor
- ☐ Dream Furniture Flipping
- ☐ Young Repurposed
- ☐ Green DIY furniture

Discount Offer

- ☐ 20% Cashback
- ☐ 5% Cashback Offer

- ☐ \$0.00 - \$150.00
- ☐ \$150.00 - \$350.00
- ☐ \$350.00 - \$504.00
- ☐ \$450.00+

\$10.00 - 20000\$

Filter by Colors


- ☐ Blue
- ☐ Orange
- ☐ Brown
- ☐ Green
- ☐ Purple
- ☐ Sky



Nautilus Lounge Chair

\$1276 \$1450


★★★★★



Alpha Chair - Solid Ebonised Oak

\$810 \$900

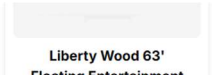
★★★★★



Replica Hans Wegner Wishbone Chair

\$675 \$750

★★★★★



Liberty Wood 63' Floating Entertainment Center

\$1012 \$1100

★★★★★

Previous

1

2





Next

2.4 Related Products in Product Detail Page

- Fetches and displays similar products based on the selected product's category or brand.

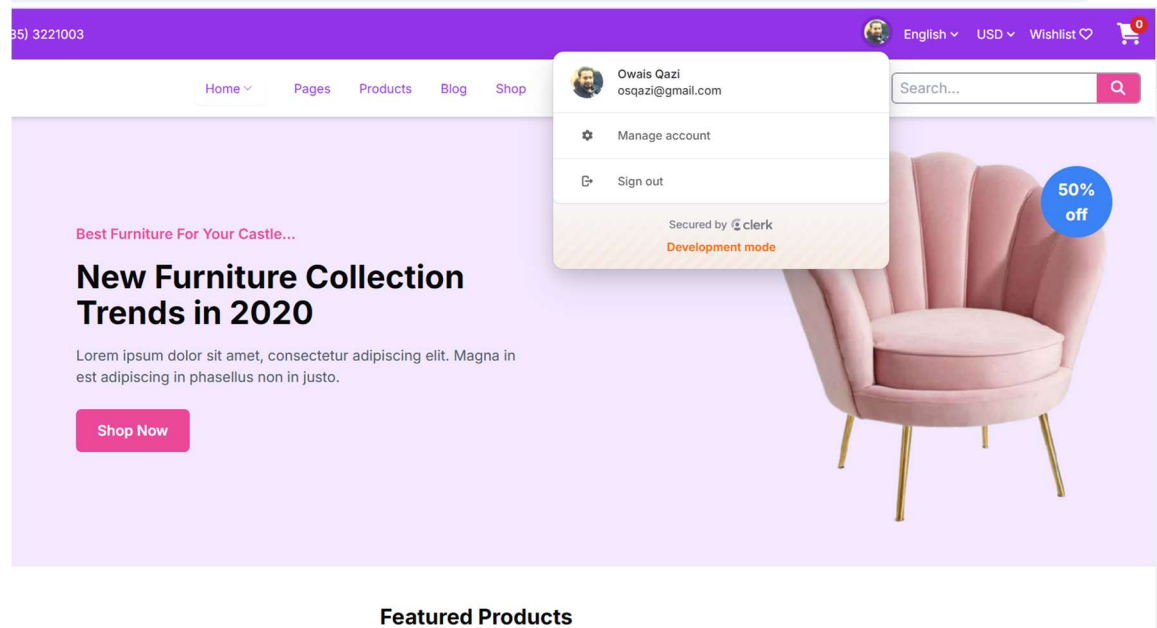
```
useEffect(() => {  
  const fetchProducts = async () => {  
    const query = `*_type == "products" && category == "${categ}" [0...4] {  
      _id,  
      name,  
      "img": image.asset->url,  
      price,  
      discountPercentage,  
      category,  
      onSale,  
      rating_5,  
      rating_4,  
      rating_3,  
      rating_2,  
      rating_1  
    }`;   
    try {  
      const data: Products[] = await client.fetch(query);  
      setProducts(data);  
    } catch (error) {  
      console.error("Error fetching products:", error);  
    }  
  }  
  fetchProducts();  
}, []);
```

Related Products


			
Uchiwa Quilted Lounge Chair ★★★★★	Rapson Thirty-Nine Guest Chair ★★★★★	Nautilus Lounge Chair ★★★★★	High Quality Modern Customized Plastic Chair ★★★★★
\$1600.00	\$1300.00	\$1450.00	\$150.00

2.5 Login Page, Cart Page, Orders History Page, Payment Page and Integration and Pagination

- Users can signup and login using their gmail or github account.
- seamless flow of product add to cart from list view and product detail view.
- Shipping Charges Calculation.
- Validation of user is logged in before placing order and proceeding payment.
- Orders Page where the logged user can see their placed order and status and if payment is pending then user can instantly make payment from this page.
- Sanity Schema to record order of authenticated users.
- Sanity Schema to maintain list of Subscribers for email alerts of offers and promotions.




Continue to Heka Furniture

 Continue with Google

or




Email address or username

Continue ▶

Secured by  clerk
Development mode

Product

PriceQuantityTotal

<div></div> <div>High Quality Modern Customized Plastic Chair Color: BROWN Size: XL</div>	150	<div>+ 2 -</div>	\$300
<div></div> <div>Nordic Net Red Chair Color: BROWN Size: XL</div>	320	<div>+ 1 -</div>	\$288
<div></div> <div>Armchair Tortuga Color: BROWN Size: XL</div>	850	<div>+ 4 -</div>	\$3400

Subtotals: \$4020

Delivery/Shipping and Taxes: \$45

Totals: \$4110

☐ Shipping and Taxes calculated at checkout

Proceed to Checkout

Update Cart

Orders History

Clear Cart

Calculate Shopping

Your Orders

Order List

Order ID: ORD-1738824941271	Order Placed on: 06/02/2025
Total Order Amount: \$2975	Order Status: Pending
Payment Status: Pending	Make Payment
Order ID: ORD-1738780417352	Order Placed on: 05/02/2025
Total Order Amount: \$1630	Order Status: Pending
Payment Status: Pending	Make Payment
Order ID: ORD-1738780812133	Order Placed on: 05/02/2025
Total Order Amount: \$678	Order Status: Pending
Payment Status: Pending	Make Payment

1 error

Order Details

Products

High Quality Modern Customized Plastic Chair
Price: \$150
Quantity: 1



Futuristic Sleek Modern Chair
Price: \$2000
Quantity: 1



Stylish Golden Metal Legs Mint Blue Fabric Velvet Sofa Leisure Armchair
Price: \$780
Quantity: 1



Please fill payment detail

 Secure, 1-click checkout with Link ▾

Card number

1234 1234 1234 1234



Expiry date

MM / YY

Security code

CVC



Country

Pakistan ▾

Pay

Hekto

Content

 Products >

 Orders >

 Subscribers >

Orders

+ ...

osqazi@gmail.com

  ... x

 Search list

 osqazi@gmail.com

 osqazi@gmail.com

 osqazi@gmail.com

Orders

osqazi@gmail.com

[Add comment](#)

Username

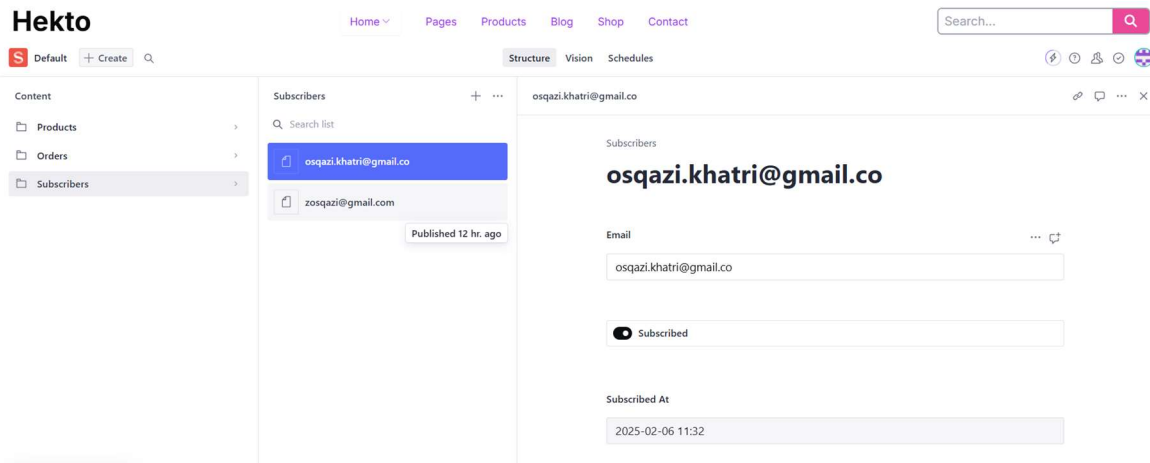
osqazi@gmail.com

First Name

owais

Last Name

qazi



3. Code Deliverables

3.1 Key Components

- ProductCard: Displays individual product details in the listing view.
- ProductList: Fetches and maps products to display using ProductCard.

```
// Generate static params
export async function generateStaticParams() {
  const query = `*[_type == "products"]{ _id }`;
  const products = await client.fetch(query);

  return products.map((product: { _id: string }) => ({
    product: product._id,
  }));
}

// Fetch product data
async function getProduct(productId: string): Promise<ProductDet> {
  'use server';

  const query = `*[_type == "products" && _id == $id][0] {
    _id,
    name,
    description,
    image{ asset->{ url } },
    href,
    price,
    discountPercentage,
    colors,
    rating_5,
    rating_4,
    rating_3,
    rating_2,
    rating_1,
    createdAt,
    category
  }`;

  return client.fetch(query, { id: productId });
}
```

- searchBar: Enables real-time product search.

```
let filteredProducts = result;
if (searchQuery) {
  filteredProducts = result.filter((product) =>
    product.name.toLowerCase().includes(searchQuery.toLowerCase())
  );
}
```

3.2 API Integration & Dynamic Routing

- Uses conventional props and params method for data fetching and routing.
- Implements API calls to Sanity CMS for fetching product details, for adding subscribers, add user orders, making payment api for integrated Stripe payment mechanism and Clerk Auth components and middleware for Authentication.
- Uses Next.js dynamic routing for handling individual product pages.

```
> api > checkout > session > TS route.ts > POST > amount
import { NextResponse } from "next/server";
import Stripe from "stripe";

const stripe = new Stripe(process.env.STRIPE_SECRET_KEY as string, {
  apiVersion: "2025-01-27.acacia",
});

export async function POST(req: Request) {
  try {
    const [{ amount }, orderId, currency] = await req.json();

    const paymentIntent = await stripe.paymentIntents.create({
      amount: amount, // Amount in cents (e.g., $10 = 1000)
      currency: currency || "usd",
      payment_method_types: ["card"],
      metadata: {orderId},
    });

    return NextResponse.json({ clientSecret: paymentIntent.client_secret });
  } catch (error: any) {
    return NextResponse.json({ error: error.message }, { status: 500 });
  }
}
```

```

middleware.ts > default > clerkMiddleware() callback
1 import { clerkMiddleware, createRouteMatcher } from '@clerk/nextjs/server'
2
3 const isProtectedRoute = createRouteMatcher(['/makepayment', '/orders']);
4
5 export default clerkMiddleware(async (auth, request) => {
6   if (isProtectedRoute(request)) {
7     await auth.protect();
8   }
9 });
10
11
12 export const config = {
13   matcher: [
14     // Skip Next.js internals and all static files, unless found in search params
15     '/(?!(next|_next|[^?]*\\.(?:html?|css|js(?!on)|jpe?g|webp|png|gif|svg|ttf|woff2?|ico|csv|docx?|xlsx?|zip|webmanifest))',
16     // Always run for API routes
17     '/(api|trpc)(.*)',
18   ],
19 }

```

```

import { NextRequest, NextResponse } from "next/server";
import { client } from "@sanity/lib/client";

export async function POST(req: NextRequest) {
  try {
    const body = await req.json(); // Parse request body

    const order = await client.create({
      _type: "order",
      ...body,
    });

    return NextResponse.json(order, { status: 201 });
  } catch (error) {
    return NextResponse.json({ error: "Failed to create order", details: error }, { status: 500 });
  }
}

```

```

import { NextRequest, NextResponse } from "next/server";
import { client } from "@sanity/lib/client";

export async function POST(req: NextRequest) {
  try {
    const body = await req.json(); // Parse request body

    if (!body.email) {
      return NextResponse.json({ error: "Email is required" }, { status: 400 });
    }

    const subscriber = await client.create({
      _type: "subscribers",
      email: body.email,
      isSubscribed: true,
      subscribedAt: new Date().toISOString(),
    });

    return NextResponse.json(subscriber, { status: 201 });
  } catch (error) {
    return NextResponse.json({ error: "Failed to subscribe", details: error }, { status: 500 });
  }
}

```

```

$ middleware.ts > default > clerkMiddleware() callback
1 import { clerkMiddleware, createRouteMatcher } from '@clerk/nextjs/server'
2
3 const isProtectedRoute = createRouteMatcher(['/makepayment', '/orders']);
4
5 export default clerkMiddleware(async (auth, request) => {
6   if (isProtectedRoute(request)) {
7     await auth.protect();
8   }
9 });
10
11
12 export const config = {
13   matcher: [
14     // Skip Next.js internals and all static files, unless found in search params
15     '/(?!(?!_next|[^?]*\\.(?:html?|css|js|?on|jpe?g|webp|png|gif|svg|ttf|woff2?|ico|csv|docx?|xlsx?|zip|webmanifest))'
16     // Always run for API routes
17     '/(api|trpc)(.*)',
18   ],
19 }

```

```

{/* Action Items Section */}
<div className="w-full lg:w-auto flex justify-between lg:justify-end items-center gap-3 mt-2 lg:mt-0">
  <span>
    {/* <Link
      href={"/login"}
      className="hover:cursor-pointer flex justify-center items-center gap-1"
    >
      Login <i className="fa-regular fa-user"></i>
    </Link>
    */}
    <SignedOut>
      <SignInButton />
    </SignedOut>
    <SignedIn>
      <UserButton />
    </SignedIn>
  </span>
  <span>
    <select className="w-auto bg-transparent focus:text-black hover:cursor-pointer">
      <option value="English" className="bg-white text-black">
        English
      </option>
    </select>
  </span>
</div>

```

4. Documentation

4.1 Steps Taken to Build & Integrate Components

- Created a Sanity CMS schema for products, orders and subscribers.
- Used a script to upload data from an external API to Sanity CMS.
- Implemented product listing and detail pages with dynamic routing.
- implemented orders history page and Payment page
- Integrated search, filters, and pagination.
- Connected Stripe for payments and Clerk for authentication.

4.2 Challenges Faced & Solutions Implemented

- Next.js 15 compatibility issues: Many features were unstable, leading to a downgrade to Next.js 14.2.2 for stable development.
- Data fetching optimization: Used conventional props and params method instead of Next.js data fetching functions.
- Styling issues with Tailwind & ShadCN UI: Adjusted styles to ensure consistency across components.

4.3 Best Practices Followed

- Followed modular component-based architecture.
- Implemented lazy loading for performance optimization.
- Used Sanity CMS for efficient content management.
- Ensured secure authentication and payment integration.

5. Conclusion

The E-Commerce Marketplace - Hekta Furniture (Template-4) successfully meets the core requirements of an e-commerce platform, offering dynamic product listings, filtering, authentication, and payment integration. Despite challenges with Next.js version compatibility, the project was optimized for performance and scalability.