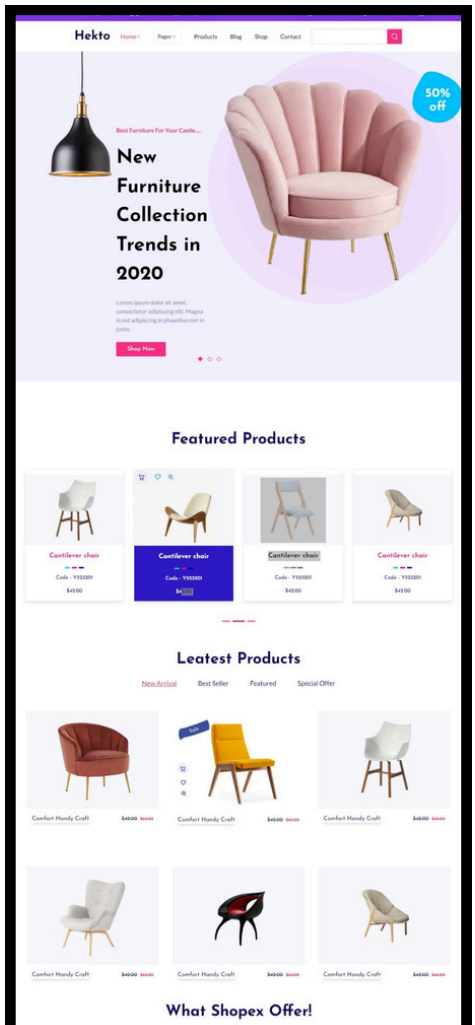
Day 4 - Dynamic Frontend Components - Haketo

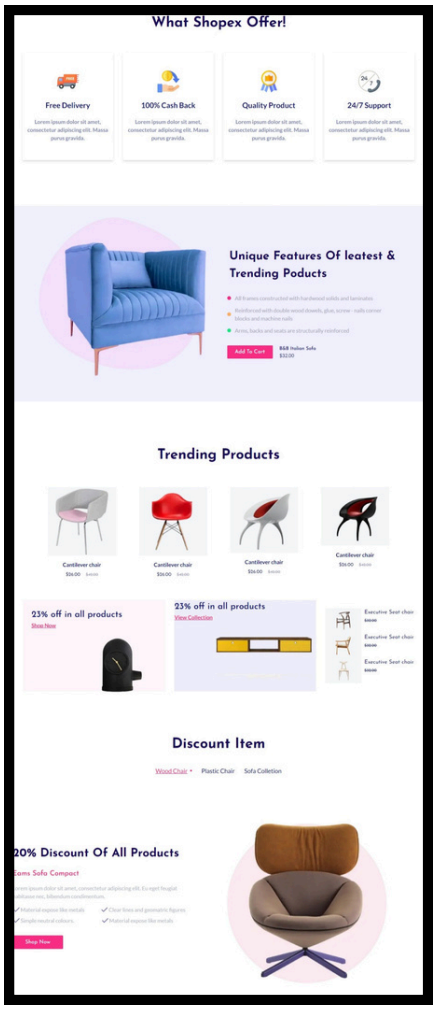
1. Functional Deliverables:

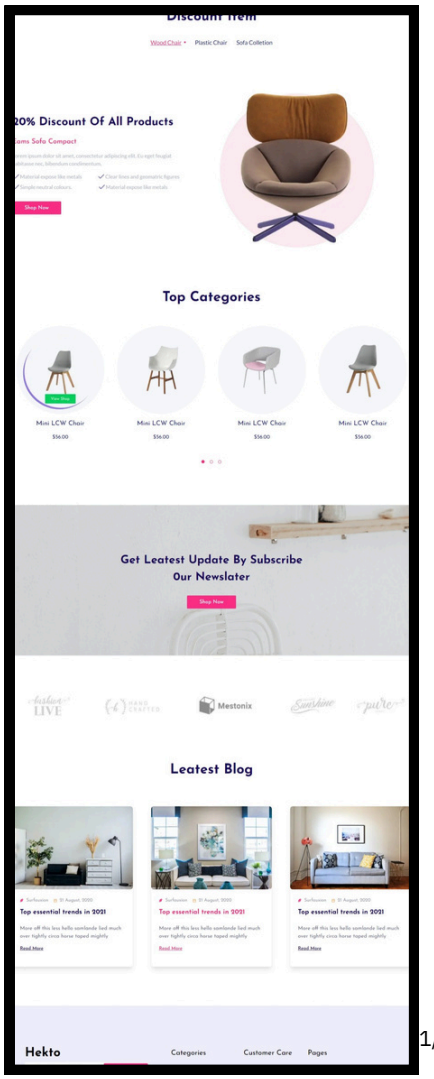
- Screenshots or screen recordings showcasing:
  - The product listing page with dynamic data. ✓
  - Individual product detail pages with accurate routing and data rendering. ✓
  - Working category filters, search bar, and pagination. (working on it)
  - Any additional features implemented, such as related products. ✓

Screenshots are available in the comparison folder inside the document folder.

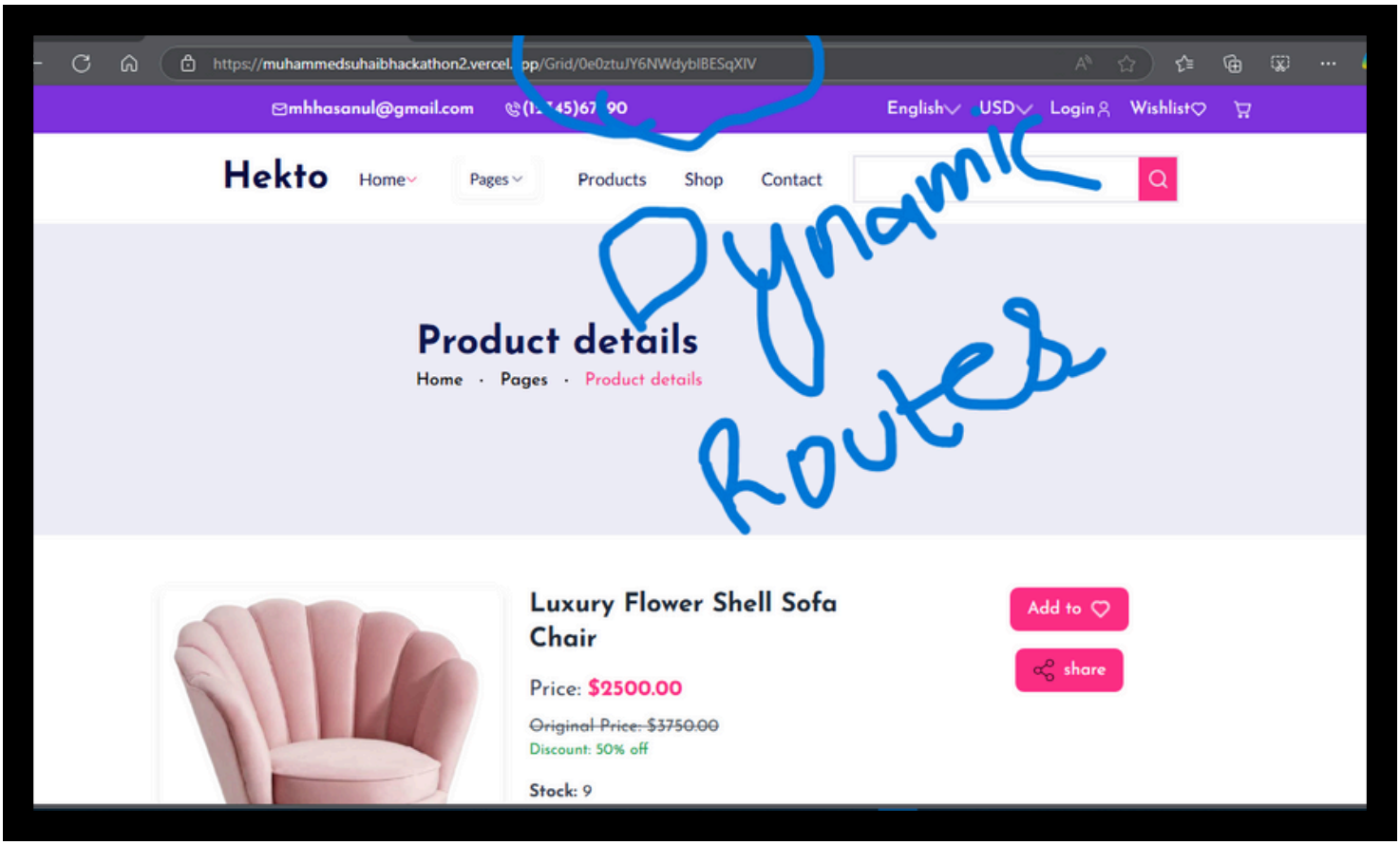
Before

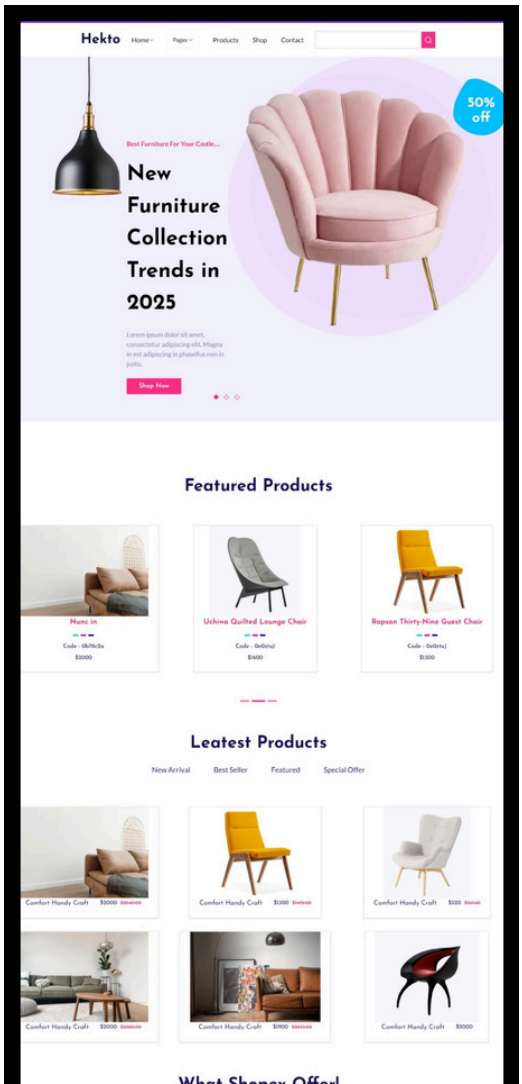


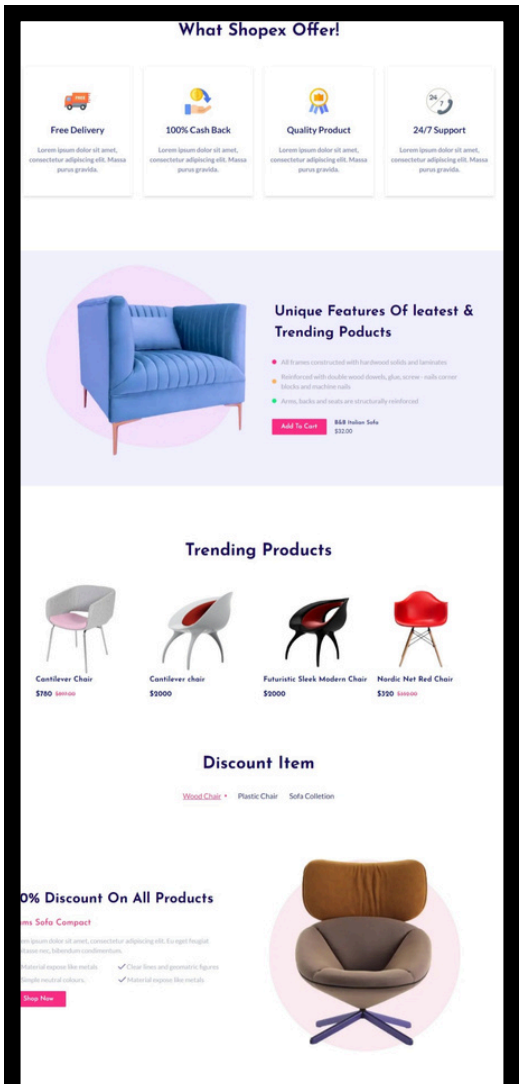


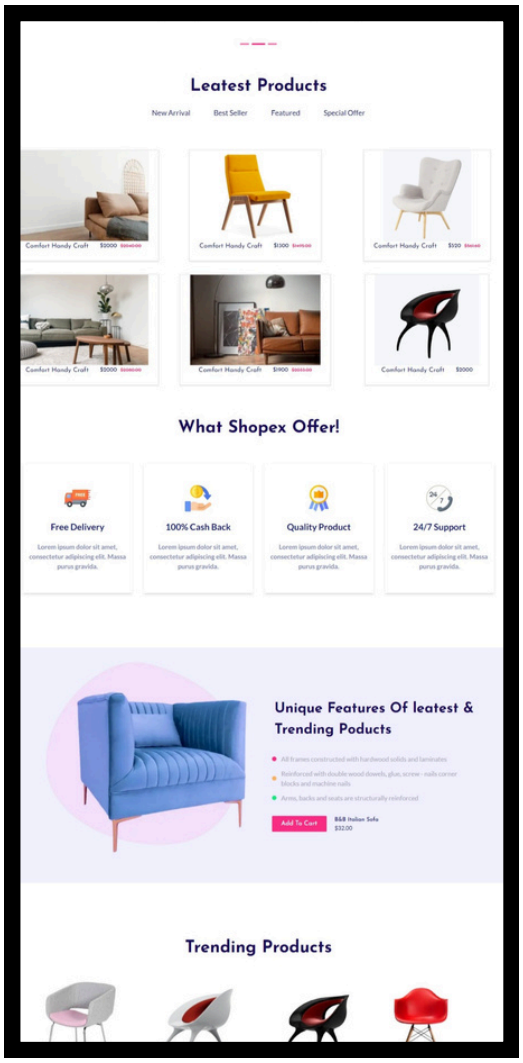


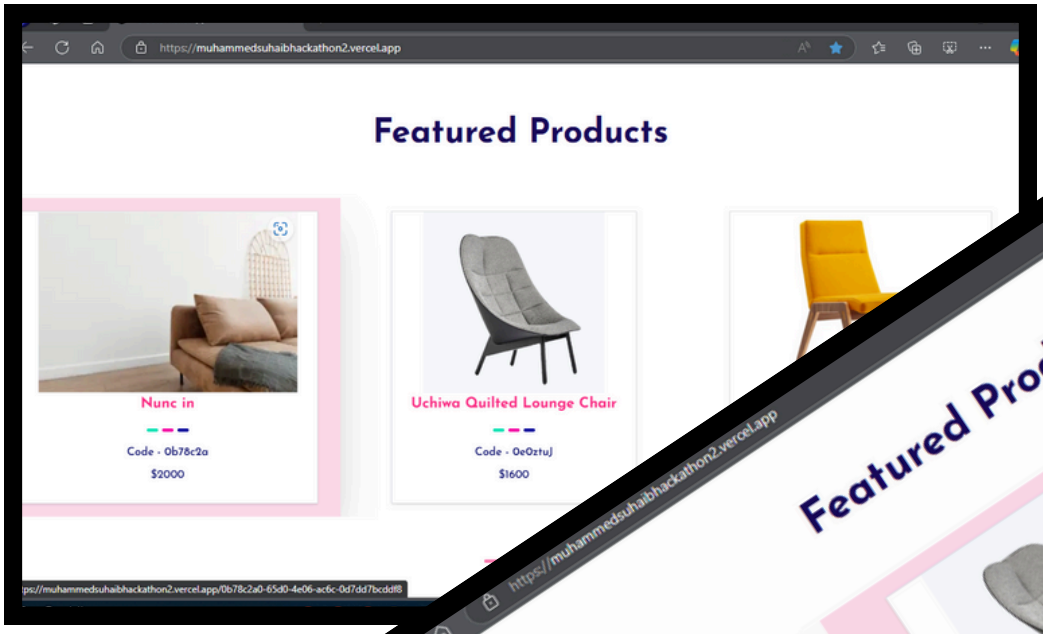
After


https://muhammedsuhaibhackathon2.vercel.app/Grid/0e0ztuJY6NWdyblBESqXIV
mhhasanul@gmail.com
English
USD
Login
Wishlist
Hekto
Home
Pages
Products
Shop
Contact
Dynamic Routes
Product details
Home · Pages · Product details
Luxury Flower Shell Sofa Chair
Price: $2500.00
Original Price: $3750.00
Discount: 50% off
Stock: 9
Add to
share


Hekto
50% off
New Furniture Collection Trends in 2025
Featured Products
Leatest Products
New Arrival
Best Seller
Featured
Special Offer

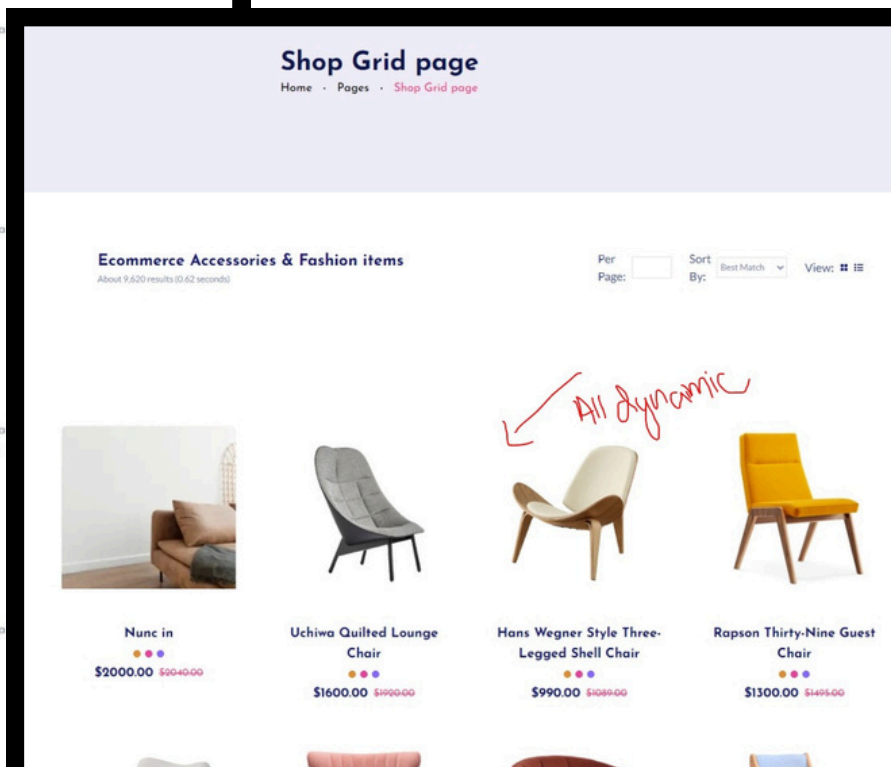

What Shopex Offer!
Free Delivery
100% Cash Back
Quality Product
24/7 Support
Unique Features Of leatest & Trending Poducts
Trending Products
Discount Item


Leatest Products
What Shopex Offer!
Free Delivery
100% Cash Back
Quality Product
24/7 Support
Unique Features Of leatest & Trending Poducts
Trending Products


https://muhammedsuhaibhackathon2.vercel.app
Featured Products
Nunc in
Code - 0b78c2a
$2000
Uchiwa Quilted Lounge Chair
Code - 0e0ztuj
$1600

Featured Products
https://muhammedsuhaibhackathon2.vercel.app
Uchiwa Quilted Lounge Chair
Code - 0e0ztuj
$1600
Rapson Thirty-Nine Gu

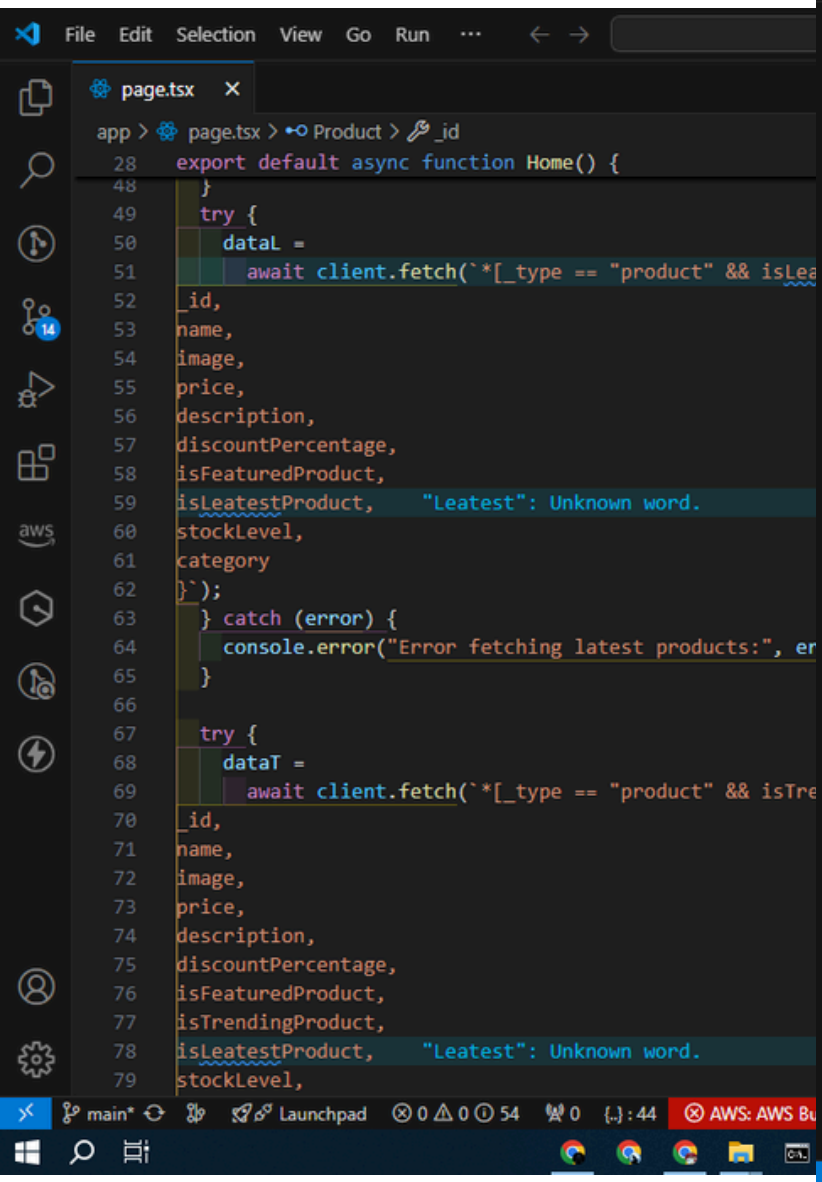
Nunc in
$2000
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Magna in est adipiscing in phasellus non in justo
All products are dynamic
Curabitur lectus
$540
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Magna in phasellus non in justo
Vitae facilisis
$540
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Magna in phasellus non in justo
In nulla
$2300
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Magna in phasellus non in justo., 4, Accumsan tincidunt, 2000, 6, Sofa
Vel sem
$1900
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Magna in phasellus non in justo

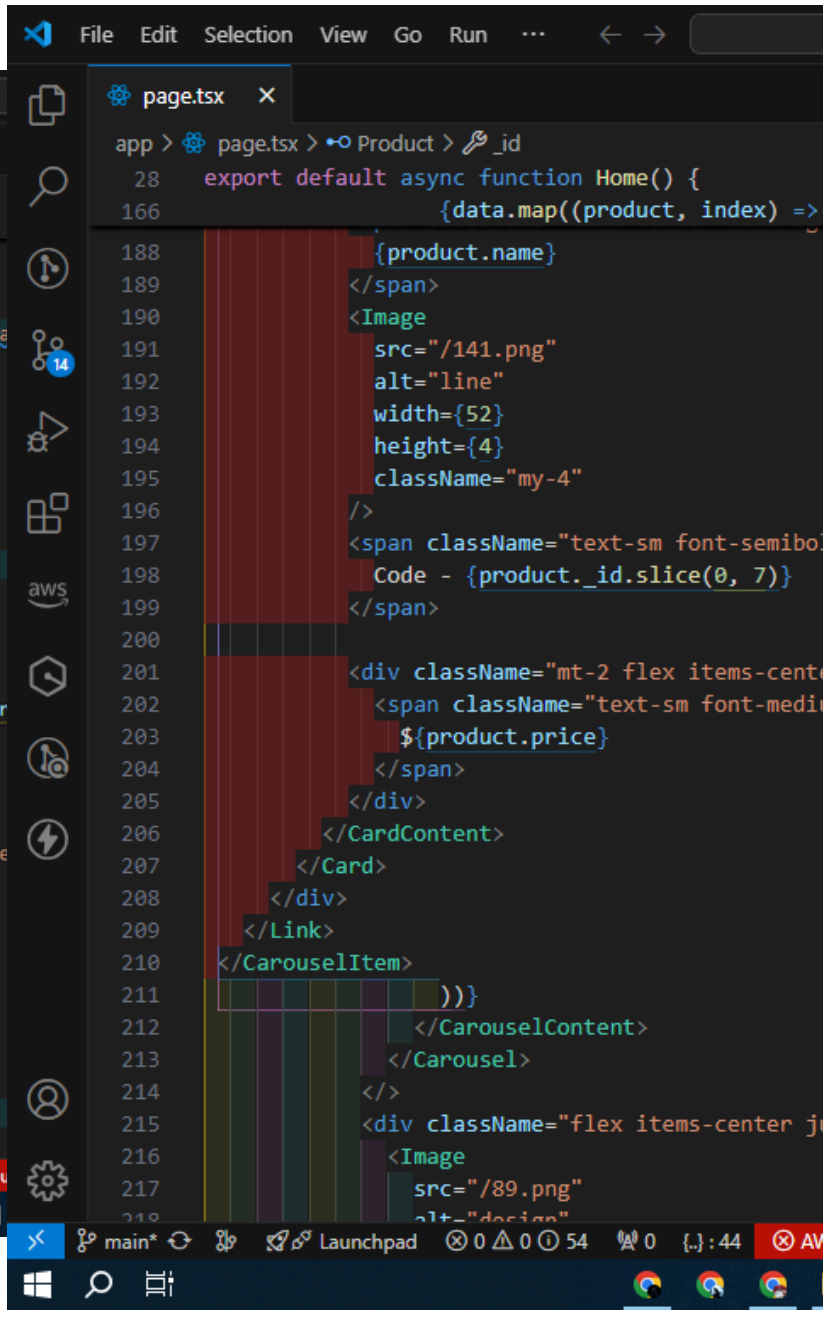

Shop Grid page
Home · Pages · Shop Grid page
Ecommerce Accessories & Fashion items
About 9,620 results (0.62 seconds)
Per Page:
Sort By:
Best Match
View:
All dynamic
Nunc in
$2000.00 $2040.00
Uchiwa Quilted Lounge Chair
$1600.00 $1990.00
Hans Wegner Style Three-Legged Shell Chair
$990.00 $1089.00
Rapson Thirty-Nine Guest Chair
$1300.00 $1495.00

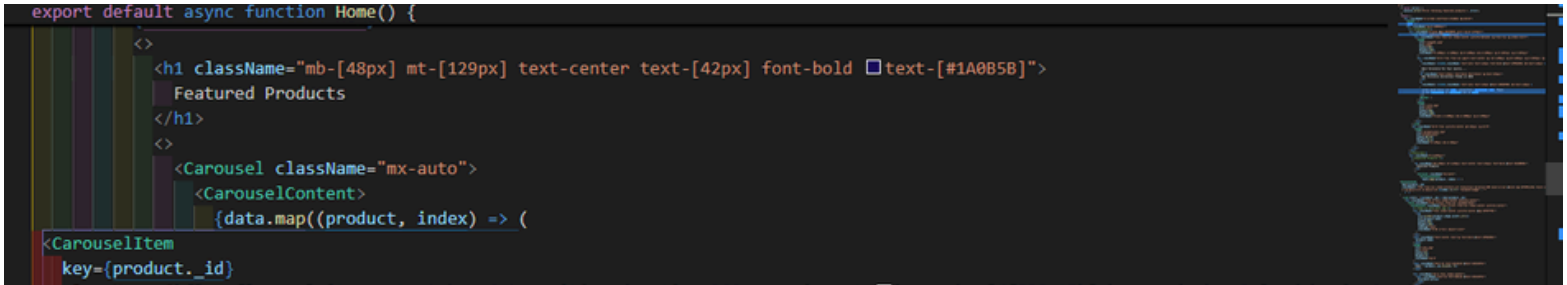
2. Code Deliverables:

- Code snippets for key components:

1. ProductCard







```
{data.map((product, index) => (
<CarouselItem
  key={product._id}
  className={`flex flex-col items-stretch p-4 transition duration-300 ease-in-out hover:bg-[#f599c15b] hover:shadow-2xl md:basis-1/
  2 lg:basis-1/3 xl:basis-1/4 ${index === 0 ? "animate-nudge" : ""
  } `}
>
  <Link href={`/${product._id}`} key={product._id}>
    <div className="flex h-full items-center justify-center">
      <Card className="flex h-full flex-col rounded-none">
        <CardContent className="flex h-full flex-col items-center justify-center">
          {/* head of card with img */}
          <div className="flex items-center justify-center bg-[#F6F7FB]">
            <img
              src={urlFor(product.image.asset).url()}
              alt={product.name}
              width={201}
              height={201}
              loading="lazy"
              className="h-56 w-full object-cover"
            />
          </div>
          <span className="text-center text-lg font-bold text-[#FB2E86]">
            {product.name}
```

2. ProductList

```
async function ShopList() {
  {/* list Section */}
  <div className="flex flex-col gap-8" >
    {dataLP.map((product) => (
      <Link
        href={`/Grid/${product._id}`}
        key={product._id}
      >
        <div key={product._id} className="flex
        flex-col md:flex-row px-[4vh] md:px-0
        items-center justify-center gap-8">
          <div>
            <img
              src={urlFor(product.image.asset).url()}
              alt={product.name}
              width={313.63}
              height={217.56}
              className="h-[217.56px] w-[313.63px]
              rounded-[10px] object-cover"
            />
          </div>
          <div className="px-[4vh] md:px-0">
            <div className="flex items-center gap-5
            text-[19px] font-semibold text-[#111C85]
            ">
              {product.name}
              <img
                src="/Group44.png"
                alt="circles"
                width={42}
                height={10}
                className="h-[10px] w-[42px]"
```

```
async function ShopList() {
  {dataLP.map((product) => (
    <span className="ml-1 text-xs text
    [#fb2e2e] line-through">
      $
      {(
        parseFloat(product.price) *
        (1 + product.
        discountPercentage / 100)
      ).toFixed(2)}
    </span>
  )}
  </span>{" "}
  <img src="/s.png" alt="5s" width={70}
  height={17.25} />
</div>
<p
  className="md:w-[591px] text-base
  font-normal text-[#9295AA]"
>
  {product.description}
</p>
<img
  src="/G35.png"
  alt="grp"
  width={144.67}
  height={34.23}
  className="-ml-5"
/>
</div>
</div>
</Link>
```

3. SearchBar

```tsx
"use client";

import Bredcrumb from "@/components/Bredcrumb";
import Footer from "@/components/Footer";
import Header from "@/components/Header";
import Mi from "@/components/Mi";
import Nvbr from "@/components/Navbar";
import { urlFor } from "@/sanity/lib/image";
import Link from "next/link";
import { useSearchParams } from "next/navigation";
import { useEffect, useState, Suspense } from "react";
import { Product } from "../Grid/page";

function SearchPage() {
  const searchParams = useSearchParams();
  const query = searchParams.get("query") || "";
  const [results, setResults] = useState<Product[]>([]);
  const [loading, setLoading] = useState(false);

  useEffect(() => {
    if (query) {
      const fetchResults = async () => {
        setLoading(true);
        try {
          const res = await fetch(`/api/search?query=${encodeURIComponent(query)}`);
          const data = await res.json();
          setResults(data.products || []);
        } catch (error) {
          console.error("Error fetching search results:", error);
          setResults([]);
        } finally {
          setLoading(false);
        }
      };

      fetchResults();
    }
  }, [query]);

  return (
    <div>
      <Header />
      <Nvbr />
      <Bredcrumb pageName="Search Results" />
      <div className="px-6 py-8">
        <h1 className="text-3xl font-bold text-gray-800 text-center mb-6">
          Search Results for &quot;{query}&quot;
        </h1>

        {loading && (
          <div className="flex justify-center items-center h-40">
            <p className="text-lg text-gray-500">Loading...</p>
          </div>
        )}

        {!loading && results.length > 0 ? (
          <div className="grid grid-cols-1 gap-8 sm:grid-cols-2 lg:grid-cols-4  ">
            {results.map((product) => (
              <div
                key={product._id}
                className="relative flex flex-col items-center rounded-lg bg-white p-4 shadow-xl shadow-[#9950f896] transition-transform hover:-translate-y-1 hover:shadow-inner  hover:shadow-[#
              >
                <Link href={`/${product._id}`}>
                  {/* Product Image */}
                  <img
                    src={product.image ? urlFor(product.image).url() : "/placeholder.png"}
                    alt={product.name}
                    className="h-48 w-full rounded-lg object-cover"
                    loading="lazy"
                  />
                </Link>
                <div className="mt-4 text-center">
                  <h3 className="text-lg font-bold text-gray-800">{product.name}</h3>
                  <p className="mt-2 text-gray-700">${parseFloat(product.price).toFixed(2)}</p>
                </div>
                <Link href={`/${product._id}`}>
                  <button className="mt-4 w-full rounded-lg bg-[#FB2E86] px-4 py-2 text-white transition hover:bg-[#e02174]">
                    View Details
                  </button>
                </Link>
              </div>
            ))}
          </div>
        ) : (
          !loading && (
            <div className="flex justify-center items-center h-40">
              <p className="text-lg text-gray-500">No results found.</p>
            </div>
          )
        )}
      </div>
      <Footer />
      <Mi />
    </div>
  );
}

export default function SearchWrapper() {
  return (
    <Suspense fallback={<p>Loading search...</p>}>
      <SearchPage />
    </Suspense>
  );
```

File Edit Selection View Go ···

route.js

app > api > search > route.js > GET

You, 22 hours ago | 1 author (You)

```js
import { client } from "@/sanity/lib/client";

export async function GET(req) {
  const url = new URL(req.url);
  const query = url.searchParams.get("query");

  if (!query) {
    return new Response(
      JSON.stringify({ error: "Query is required" }),
      { status: 400 }
    );
  }

  try {
    const results = await client.fetch(
      `*[_type == "product" && name match $query + "*"]{
        _id,
        name,
        description,
        price,
        category,
        image

      }`,
      { query }
    );


    return new Response(
      JSON.stringify({ products: results || []}),
      { status: 200 }
    );
  } catch (error) {
    console.error("Sanity fetch error:", error);
    return new Response(
      JSON.stringify({ error: "Failed to fetch data" }),
      { status: 500 }
    );
  }
```

main* Launchpad ⊗0 ⚠0

```jsx
{/* Search bar */}
<div className="flex items-center justify-between border-[2px] rounded-[7px] border-[#E7E6EF] sm:justi
  <div className="w-full px-4">
    <input
      type="text"
      value={query}
      placeholder="Search"
      onChange={handleChange} // Capture search input
      onKeyDown={(e) => {
        if (e.key === "Enter") {
          handleSearch(); // Trigger search on Enter key press
        }
      }}
      className={`${lato.className} w-full lg:pl-[10px] 2xl:pl-[110px] `}
    />
  </div>
  <div
    onClick={handleSearch} // Trigger search on click
    className="flex h-10 w-11 items-center justify-center rounded-[4px] bg-[#FB2E86]"
  >
    <Image
      src="/uil_search.png"
      alt="Search Icon"
      width={20}
      height={20}
    />
  </div>
</div>
```

Scripts or logic for API integration and dynamic routing:

```tsx
import Footer from "@/components/Footer";
import Header from "@/components/Header";
import Mi from "@/components/Mi";
import Nvbr from "@/components/Navbar";
import { client } from "@/sanity/lib/client";
import { urlFor } from "@/sanity/lib/image";
import Image from "next/image";
import Link from "next/link";

interface Product {
  _id: string;
  name: string;
  image: { asset: { url: string } } | null;
  price: string;
  description: string;
  discountPercentage: number;
  isFeaturedProduct: boolean;
  stockLevel: number;
  category: "Chair" | "Sofa";
}

interface Params {
  params: {
    id: string;
  };
}

async function page({ params }: Params) {
  const resp: Product = await client.fetch(
    `*[_type == "product" && _id == $id][0]{
        name,
        image,
        price,
        description,
        discountPercentage,
        isFeaturedProduct,
        stockLevel,
        category
    }`,
    { id: params.id },
  );

  const similarProducts: Product[] = await client.fetch(
    `*[_type == "product" && category == $category && _id != $id]{
        _id,
        name,
        image,
        price,
        discountPercentage
    }`,
    { category: resp.category, id: params.id },
  );

  return (
    <div>
      <Header />
      <Nvbr />
      <Bredcrumb pageName="Product details" />
      <div className="container mx-auto my-12 px-4">
        {/* Product Section */}
        <div className="flex flex-col gap-6 md:flex-row">
          {/* Left: Product Image */}
          <div className="flex-1">
            <img
              src={resp.image ? urlFor(resp.image).url() : "/placeholder.png"}
              alt={resp.name}
              className="size-[325px] rounded-lg object-cover shadow-md"
            />
          </div>

          {/* mid: Product Details */}
          <div className="flex-1">
            <h1 className="mb-4 text-2xl font-bold text-gray-800">
              {resp.name}
            </h1>
            <p className="mb-2 text-xl text-gray-700">
              Price:{" "}
              <span className="font-bold text-[#FB2E86]">
                ${parseFloat(resp.price).toFixed(2)}
              </span>
            </p>
            {resp.discountPercentage > 0 && (
              <p className="text-md text-gray-600 line-through">
```

```
    id: string;
  };
}

async function page({ params }: Params) {
  const resp: Product = await client.fetch(
    `*[_type == "product" && _id == $id][0]{
        name,
        image,
        price,
        description,
        discountPercentage,
        isFeaturedProduct,
        stockLevel,
        category
    }`,
    { id: params.id },
  );

  const similarProducts: Product[] = await client.fetch(
    `*[_type == "product" && category == $category && _id != $id]{
        _id,
        name,
        image,
        price,
        discountPercentage
    }`,
    { category: resp.category, id: params.id },
  );

  return (
    <div>
      <Header />
      <Nvbr />
      <Bredcrumb pageName="Product details" />
      <div className="container mx-auto my-12 px-4">
        {/* Product Section */}
        <div className="flex flex-col gap-6 md:flex-row">
          {/* Left: Product Image */}
          <div className="flex-1">
            <img
              src={resp.image ? urlFor(resp.image).url() : "/placeholder.png"}
              alt={resp.name}
              className="size-[325px] rounded-lg object-cover shadow-md"
            />
          </div>

          {/* mid: Product Details */}
          <div className="flex-1">
            <h1 className="mb-4 text-2xl font-bold text-gray-800">
              {resp.name}
            </h1>
            <p className="mb-2 text-xl text-gray-700">
              Price:{" "}
              <span className="font-bold text-[#FB2E86]">
                ${parseFloat(resp.price).toFixed(2)}
              </span>
            </p>
            {resp.discountPercentage > 0 && (
              <p className="text-md text-gray-600 line-through">
                Original Price: $
                {(
                  parseFloat(resp.price) *
                  (1 + resp.discountPercentage / 100)
                ).toFixed(2)}
              </p>
            )}
            <p className="mb-4 text-sm text-green-600">
              Discount: {resp.discountPercentage}% off
            </p>
            <p className="text-gray-700">
              <span className="font-bold">Stock:</span> {resp.stockLevel}
            </p>
            <p className="text-gray-700">
              <span className="font-bold">Category:</span> {resp.category}
            </p>
            <p className="mt-4 text-gray-600">{resp.description}</p>
            {/* Buttons */}
            <div className="mt-6 flex gap-4">
              <button className="mx-auto flex h-[39px] w-full rounded-lg bg-[#FB2E86] px-6 py-2 font-medium text-white transition duration-300 hover:scale-105 hover:
                Add to
                <Image
                  width={20}
                  height={20}
                  src="/crt.png"
                  alt="Wishlist"
                  className="h-5 w-5"
                />
              </button>

              <button className=" bg-gray-200 rounded-lg  flex px-6  mx-auto h-[39px] w-full   font-medium  duration-300 hover:scale-105 focus:outline-none focus:ri
                Buy Now
```

```
            <button className="bg-gray-200 rounded-lg flex px-6 mx-auto h-[39px] w-full font-medium duration-500 hover:scale-105 focus:outline-none focus:ri
              Buy Now
            </button>
          </div>
        </div>
        {/* Right:  Buttons */}
        <div className="flex flex-1 flex-row items-center gap-4 md:flex-col">
          <button className="flex items-center gap-2 rounded-lg bg-[#FB2E86] px-4 py-2 text-white transition hover:bg-[#e02174]">
            Add to
            <Image
              width={20}
              height={20}
              src="/uil_heart-alt.png"
              alt="Wishlist"
              className="h-5 w-5"
            />
          </button>
          <button className="flex items-center gap-2 rounded-lg bg-[#FB2E86] px-4 py-2 text-white transition hover:bg-[#e02174]">
            <Image
              width={20}
              height={20}
              src="/share.png"
              alt="Wishlist"
              className="h-5 w-5"
            />
            share
          </button>
        </div>
      </div>

      {/* Product Details Section */}
      <div className="mt-12 rounded-lg bg-gray-100 p-4">
        <h2 className="mb-4 text-xl font-bold text-gray-800">
          Product Details
        </h2>
        <p className="text-gray-700">{resp.description}</p>
      </div>

      {/* Similar Products Section */}
      <div className="mt-12">
  <h2 className="mb-6 text-xl font-bold text-gray-800">Similar Products</h2>
  <div className="grid grid-cols-1 gap-6 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4">
    {similarProducts.slice(0, 4).map((product) => (
      <div
        key={product._id}
        className="flex flex-col items-center rounded-lg bg-white p-4 shadow-md transition hover:shadow-xl"
      >
        <img
          src={
            product.image
              ? urlFor(product.image).url()
              : "/placeholder.png"
          }
          alt={product.name}
          className="h-48 w-full rounded-t-lg object-cover"
        />
        <h3 className="mt-4 text-lg font-bold text-gray-800">
          {product.name}
        </h3>
        <p className="text-gray-700">
          Price: ${parseFloat(product.price).toFixed(2)}
        </p>
        {product.discountPercentage > 0 && (
          <p className="text-sm text-gray-500 line-through">
            $
            {(
              parseFloat(product.price) *
              (1 + product.discountPercentage / 100)
            ).toFixed(2)}
          </p>
        )}

        <Link
          href={`/${product._id}`}
          className="mt-4 rounded-lg bg-[#FB2E86] px-4 py-2 text-white transition hover:bg-[#e02174]"
        >
          View Details
        </Link>
      </div>
    ))}
  </div>
</div>

      </div>
      <Footer />
      <Mi />
    </div>
  );
}

export default page;
here
```

Completed Tasks from Day 2:

⭐ Integrated a carousel from ShadCN ✓
⭐Added additional tags in the schema, such as "Featured," "Latest Products," "Trending," etc., along with extra details ✓
⭐Added more products ✓
⭐Prepared necessary data for frontend functionality ✓

Page Name	Static	Dynamic
Home	✓	✓
Listing Product	✓	✓
Product Details	✓	✓
Cart	✓	✓
Checkout	✓	✓
order	✓	✓
order complete	✓	✓

Day 4 Progress:

⭐Successfully rendered and displayed data on the UI without issues.
⭐ Added effects to components to give a live feel, e.g., on logos and buttons.
♦ Attempted to create a reusable component for dynamic routes, but it required the page to be client-side. Eventually, abandoned the approach and handled it manually.
🔍Searched about Emails.js for contact us options
🚨 .env*.local was already present in .gitignore, but it still uploaded my .env file to GitHub. So, I added this line manually

below the existing ones:

```
# Local env files
.env*.local
.env
```

When I tried to delete .env from GitHub, there was no option to delete it. I did a silly thing—edited .env directly on GitHub, removed all credentials, and wrote only a slash (/) before saving it. Then I thought about uploading my credentials to Vercel, but the project was running fine without it.
I didn't pull those changes to my local project. Instead, I deleted .env locally, pushed the changes, created a new .env, and added variables again. Finally, at the deployment step, I got the error I had been waiting for 🧐. I uploaded the variables on Vercel, and now everything is fine.
⛔ However, the variables are still visible in GitHub's history. So, I decided that when I sell this project, I'll create a new repo with new Sanity variables.
⛔ Fixed the navbar and tried to add active effects instead of hover, but it didn't work. I tried using the ShadCN menu navigation, but it didn't suit my project, so I left it as it was and started focusing on functionality.
🛒searched how to add "**add to cart**" functionality, but everything went over my head because my project structure is different from others. I tried taking help from GPT, but it used the Context API, and I don't have the time to learn that from scratch.
I tried doing it with my JavaScript programming skills, but I got stuck. After sending my code to GPT, I finally got it working. It took 5-6 hours, but in the end,
🤍 also implemented a **wishlist** functionality and created a wishlist page.

🗒️ Adding list & grid mode
⭐ Added search functionality
⭐ Sorting feature
⭐ Pagination
🛒 Fixed some issues with the cart feature
🤍 Fixed wishlist issues too
⭐ Migrate to pnpm
⭐ Add share button
⭐ Ensure and check all code follows clean code principles
⭐ Debug search feature
⭐ Implement "Not Found" page
⭐ Add meta title & description
⭐ Use Suspense tag
⭐ Add placeholder
⭐ Design a nice UI
⭐ Shadcn ui Skeleton
⭐ Proceed to checkout page