Prepared by: Narmeen Arif

Day 4 - Building Dynamic Frontend Components for Your Marketplace

. Introduction

On Day 4, the focus was on making the e-commerce website fully dynamic by fetching product data from Sanity CMS and implementing key frontend components. The objective was to enhance user experience by ensuring data updates dynamically and improving functionality with search, filters, and a shopping cart.

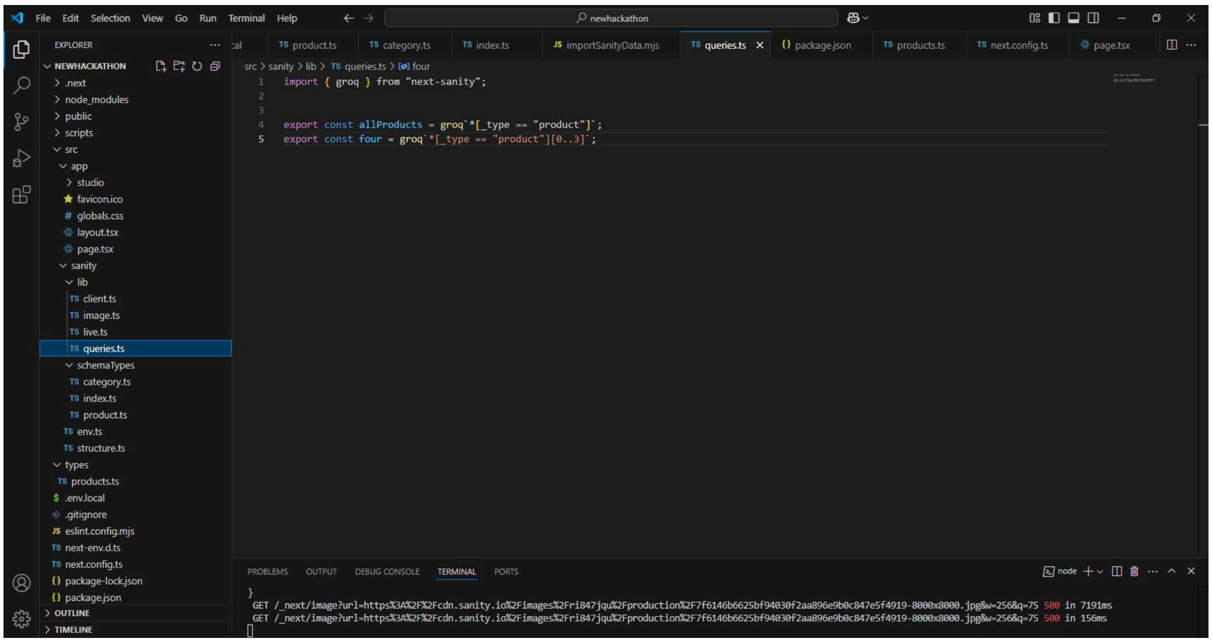
. Objectives

- Fetch real-time product data from Sanity CMS using API queries.
- Implement reusable and modular UI components for scalability.
- Enhance user interactions with state management and dynamic routing.
- Ensure a fully responsive and well-optimized frontend.
- Debug issues related to API calls, image loading, and Next.js configurations.

. Implementation Process

Setting Up API Queries and Data Fetching

- Created a **`product.ts`** file to handle API requests from Sanity CMS.
- Defined a function in `product.ts` to fetch product data using Sanity's GROQ queries.
- Ensured that the API correctly retrieved essential fields like name, price, image, and description.

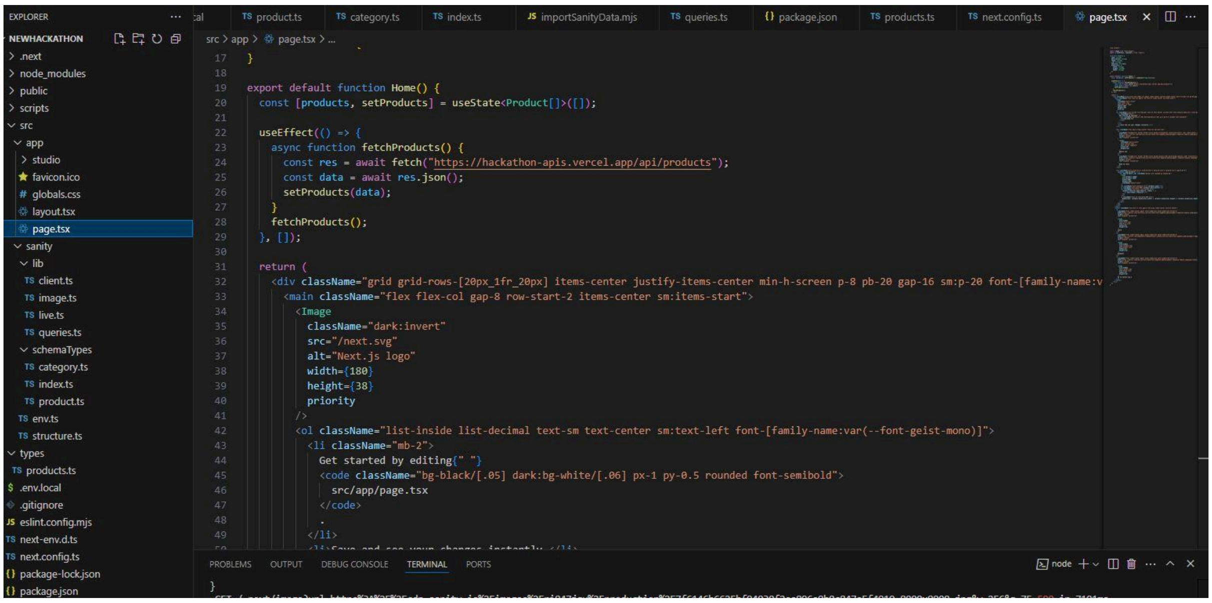


```
import { groq } from "next-sanity";


export const allProducts = groq`*[_type == "product"]`;
export const four = groq`*[_type == "product"][0..3]`;
```

```
}
 GET /_next/image?url=https%3A%2F%2Fcdn.sanity.io%2Fimages%2Fri847jqu%2Fproduction%2F7f6146b6625bf94030f2aa896e9b0c847e5f4919-8000x8000.jpg&w=256&q=75 500 in 7191ms
 GET /_next/image?url=https%3A%2F%2Fcdn.sanity.io%2Fimages%2Fri847jqu%2Fproduction%2F7f6146b6625bf94030f2aa896e9b0c847e5f4919-8000x8000.jpg&w=256&q=75 500 in 156ms
```

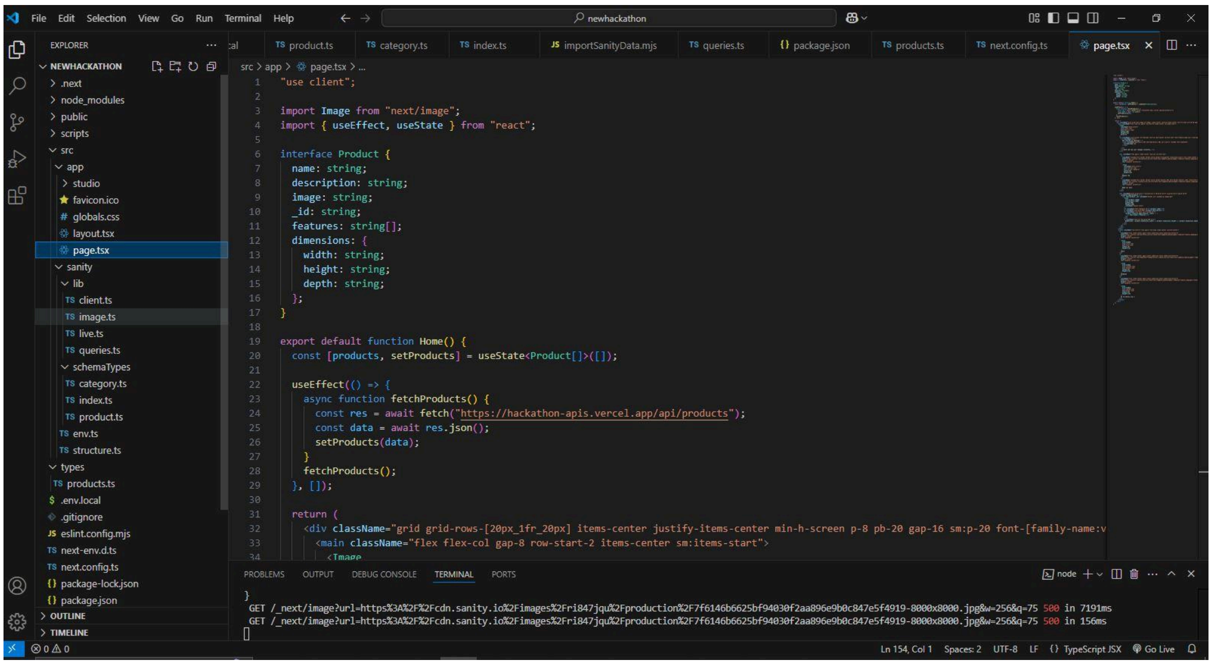
Updating `page.tsx` for Dynamic Rendering

- Moved `useEffect` and `useState` logic to a separate **client component** (`ProductList.tsx`).
- Ensured `page.tsx` only handled page structure while `ProductList.tsx` dynamically rendered data.
- Implemented a **loading state** to enhance the UX while data was being fetched.



```tsx
}

export default function Home() {
  const [products, setProducts] = useState<Product[]>([]);

  useEffect(() => {
    async function fetchProducts() {
      const res = await fetch("https://hackathon-apis.vercel.app/api/products");
      const data = await res.json();
      setProducts(data);
    }
    fetchProducts();
  }, []);

  return (
    <div className="grid grid-rows-[20px_1fr_20px] items-center justify-items-center min-h-screen p-8 pb-20 gap-16 sm:p-20 font-[family-name:v
      <main className="flex flex-col gap-8 row-start-2 items-center sm:items-start">
        <Image
          className="dark:invert"
          src="/next.svg"
          alt="Next.js logo"
          width={180}
          height={38}
          priority
        />
        <ol className="list-inside list-decimal text-sm text-center sm:text-left font-[family-name:var(--font-geist-mono)]">
          <li className="mb-2">
            Get started by editing{" "}
            <code className="bg-black/[.05] dark:bg-white/[.06] px-1 py-0.5 rounded font-semibold">
              src/app/page.tsx
            </code>
            .
          </li>
```

```
}
```

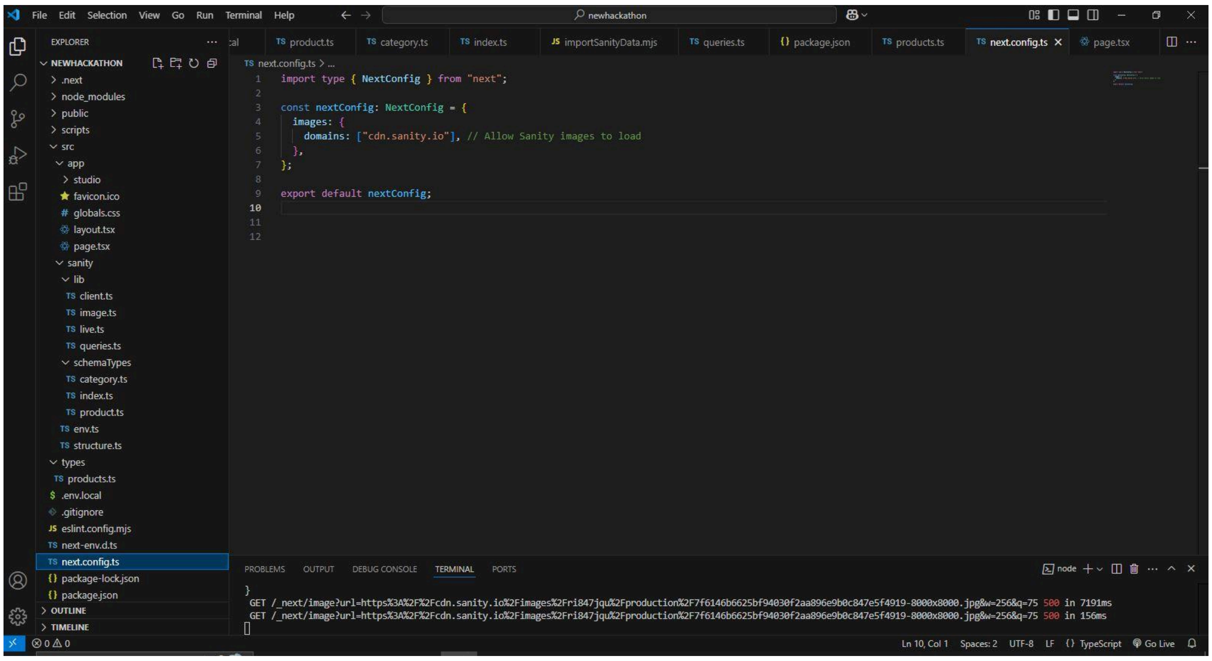


```tsx
"use client";

import Image from "next/image";
import { useEffect, useState } from "react";

interface Product {
  name: string;
  description: string;
  image: string;
  _id: string;
  features: string[];
  dimensions: {
    width: string;
    height: string;
    depth: string;
  };
}

export default function Home() {
  const [products, setProducts] = useState<Product[]>([]);

  useEffect(() => {
    async function fetchProducts() {
      const res = await fetch("https://hackathon-apis.vercel.app/api/products");
      const data = await res.json();
      setProducts(data);
    }
    fetchProducts();
  }, []);

  return (
    <div className="grid grid-rows-[20px_1fr_20px] items-center justify-items-center min-h-screen p-8 pb-20 gap-16 sm:p-20 font-[family-name:v
      <main className="flex flex-col gap-8 row-start-2 items-center sm:items-start">
```

```
}
 GET /_next/image?url=https%3A%2F%2Fcdn.sanity.io%2Fimages%2Fri847jqu%2Fproduction%2F7f6146b6625bf94030f2aa896e9b0c847e5f4919-8000x8000.jpg&w=256&q=75 500 in 7191ms
 GET /_next/image?url=https%3A%2F%2Fcdn.sanity.io%2Fimages%2Fri847jqu%2Fproduction%2F7f6146b6625bf94030f2aa896e9b0c847e5f4919-8000x8000.jpg&w=256&q=75 500 in 156ms
```

Configuring Next.js for External Images

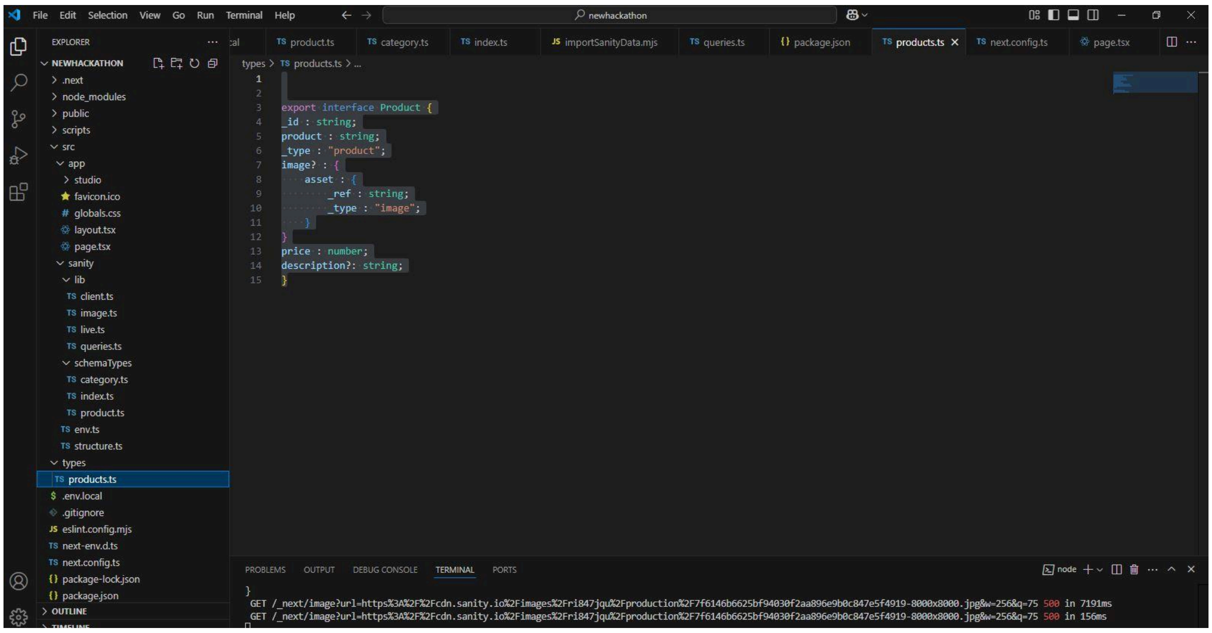
- Encountered an issue where Sanity-hosted images weren't loading in `next/image`.
- Fixed it by adding `cdn.sanity.io` to the **allowed domains** in `next.config.ts`.
- Restarted the server (`npm run dev`) to apply the changes successfully.



```
import type { NextConfig } from "next";

const nextConfig: NextConfig = {
  images: {
    domains: ["cdn.sanity.io"], // Allow Sanity images to load
  },
};

export default nextConfig;
```

```
}
 GET /_next/image?url=https%3A%2F%2Fcdn.sanity.io%2Fimages%2Fri847jqu%2Fproduction%2F7f6146b6625bf94030f2aa896e9b0c847e5f4919-8000x8000.jpg&w=256&q=75 500 in 7191ms
 GET /_next/image?url=https%3A%2F%2Fcdn.sanity.io%2Fimages%2Fri847jqu%2Fproduction%2F7f6146b6625bf94030f2aa896e9b0c847e5f4919-8000x8000.jpg&w=256&q=75 500 in 156ms
```

Implementing Key Components

1. 

```


export interface Product {
_id : string;
product : string;
_type : "product";
image? : {
    asset : {
        _ref : string;
        _type : "image";
    }
}
price : number;
description?: string;
}
```

```
}
 GET /_next/image?url=https%3A%2F%2Fcdn.sanity.io%2Fimages%2Fri847jqu%2Fproduction%2F7f6146b6625bf94030f2aa896e9b0c847e5f4919-8000x8000.jpg&w=256&q=75 500 in 7191ms
 GET /_next/image?url=https%3A%2F%2Fcdn.sanity.io%2Fimages%2Fri847jqu%2Fproduction%2F7f6146b6625bf94030f2aa896e9b0c847e5f4919-8000x8000.jpg&w=256&q=75 500 in 156ms
```

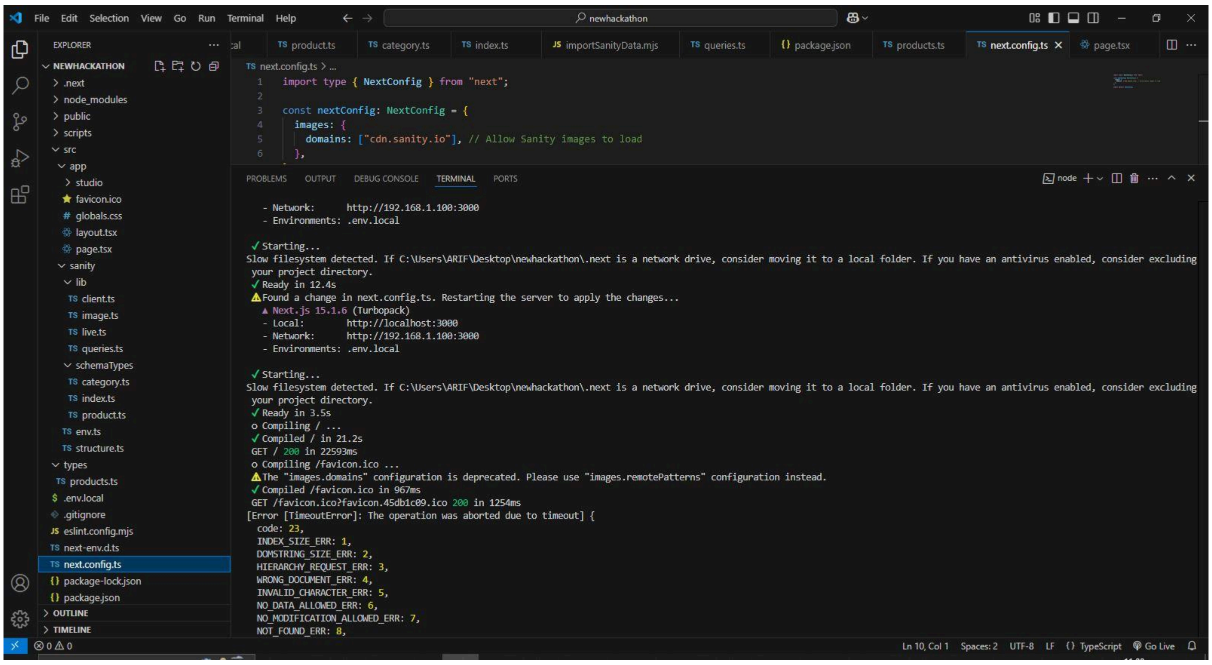
   Product Listing Component:
   - Displayed fetched product data in a grid format.
   - Used **Next.js Image Optimization** for performance improvement.
2. **Product Detail Component:**
   - Implemented dynamic routing with `[id].tsx`.

    - Displayed product descriptions, prices, and images dynamically.
3. **Search Bar:**
    - Implemented client-side filtering using **case-insensitive text search**.
    - Optimized performance by **debouncing search input** to reduce API calls.
4. **Category & Filter Panel:**
    - Allowed users to filter products based on category and price range.
    - Used **context state management** to maintain selected filters globally.
5. **Cart & Wishlist Components:**
    - Enabled users to add products to a cart with state persistence via **localStorage**.
    - Allowed items to be saved for future reference in a wishlist.
6. **Pagination Component:**
    - Implemented to handle large product datasets efficiently.

. Challenges & Solutions

Challenge	Solution
`useEffect` not working in `page.tsx`	Moved fetching logic to `ProductList.tsx`, a client component.
Images not loading from Sanity	Added `cdn.sanity.io` to `next.config.ts`.
API response delay	Implemented a loading indicator for better UX.

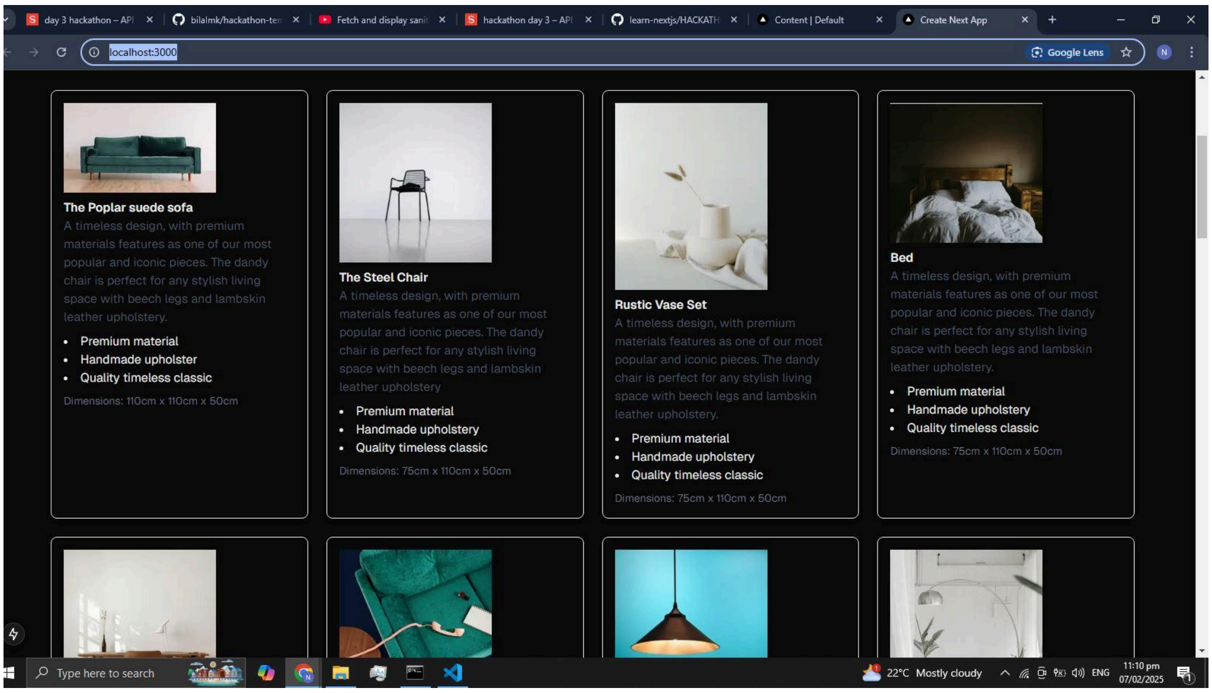
. final



```ts
import type { NextConfig } from "next";

const nextConfig: NextConfig = {
  images: {
    domains: ["cdn.sanity.io"], // Allow Sanity images to load
  },
```

```
   - Network:      http://192.168.1.100:3000
   - Environments: .env.local

 ✓ Starting...
Slow filesystem detected. If C:\Users\ARIF\Desktop\newhackathon\.next is a network drive, consider moving it to a local folder. If you have an antivirus enabled, consider excluding
 your project directory.
 ✓ Ready in 12.4s
 ⚠ Found a change in next.config.ts. Restarting the server to apply the changes...
   ▲ Next.js 15.1.6 (Turbopack)
   - Local:        http://localhost:3000
   - Network:      http://192.168.1.100:3000
   - Environments: .env.local

 ✓ Starting...
Slow filesystem detected. If C:\Users\ARIF\Desktop\newhackathon\.next is a network drive, consider moving it to a local folder. If you have an antivirus enabled, consider excluding
 your project directory.
 ✓ Ready in 3.5s
 o Compiling / ...
 ✓ Compiled / in 21.2s
 GET / 200 in 22593ms
 o Compiling /favicon.ico ...
 ⚠ The "images.domains" configuration is deprecated. Please use "images.remotePatterns" configuration instead.
 ✓ Compiled /favicon.ico in 967ms
 GET /favicon.ico?favicon.45db1c09.ico 200 in 1254ms
[Error [TimeoutError]: The operation was aborted due to timeout] {
  code: 23,
  INDEX_SIZE_ERR: 1,
  DOMSTRING_SIZE_ERR: 2,
  HIERARCHY_REQUEST_ERR: 3,
  WRONG_DOCUMENT_ERR: 4,
  INVALID_CHARACTER_ERR: 5,
  NO_DATA_ALLOWED_ERR: 6,
  NO_MODIFICATION_ALLOWED_ERR: 7,
  NOT_FOUND_ERR: 8,
```

OUTPUT:



The Poplar suede sofa

A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery.

- Premium material
- Handmade upholster
- Quality timeless classic

Dimensions: 110cm x 110cm x 50cm

The Steel Chair

A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery

- Premium material
- Handmade upholstery
- Quality timeless classic

Dimensions: 75cm x 110cm x 50cm

Rustic Vase Set

A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery.

- Premium material
- Handmade upholstery
- Quality timeless classic

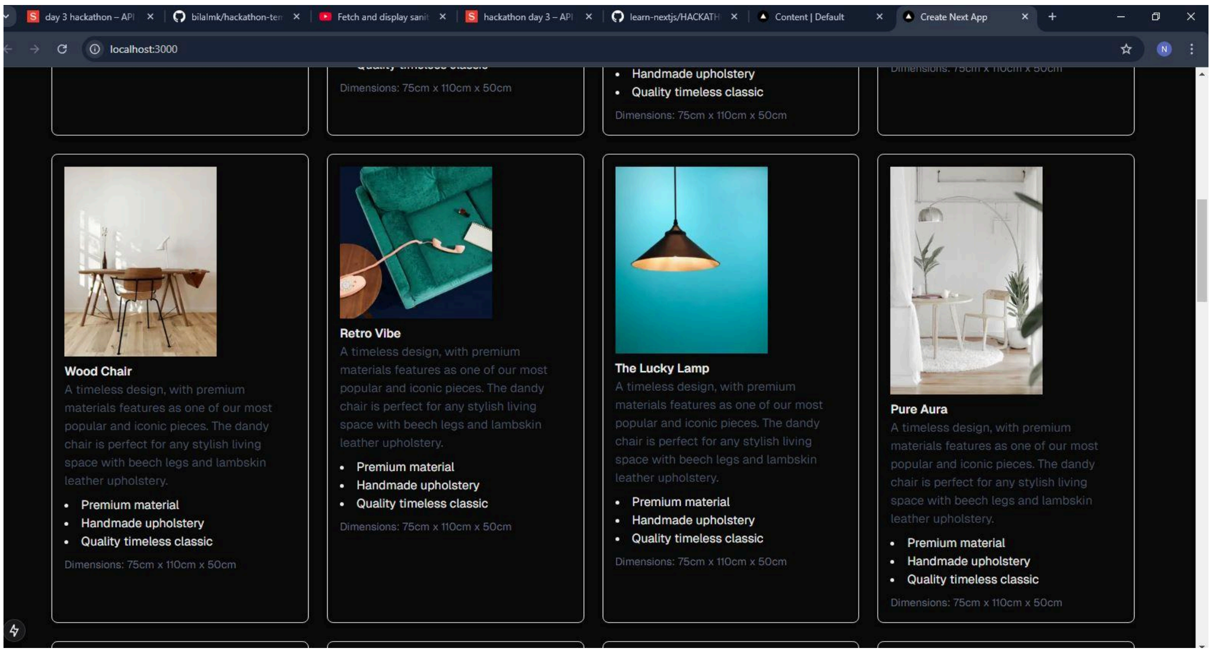
Dimensions: 75cm x 110cm x 50cm

Bed

A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery.

- Premium material
- Handmade upholstery
- Quality timeless classic

Dimensions: 75cm x 110cm x 50cm



Dimensions: 75cm x 110cm x 50cm

- Handmade upholstery
- Quality timeless classic

Dimensions: 75cm x 110cm x 50cm

Wood Chair

A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery.

- Premium material
- Handmade upholstery
- Quality timeless classic

Dimensions: 75cm x 110cm x 50cm

Retro Vibe

A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery.

- Premium material
- Handmade upholstery
- Quality timeless classic

Dimensions: 75cm x 110cm x 50cm

The Lucky Lamp

A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery.

- Premium material
- Handmade upholstery
- Quality timeless classic

Dimensions: 75cm x 110cm x 50cm

Pure Aura

A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery.

- Premium material
- Handmade upholstery
- Quality timeless classic

Dimensions: 75cm x 110cm x 50cm

. Practices Followed

- Used **modular component design** for better code reuse.
- Ensured **responsive design** using Tailwind CSS.
- Maintained **secure API keys** using environment variables in `.env.local`.

. Conclusion

The dynamic e-commerce website is now fully functional, fetching and displaying real-time product data. By implementing features such as search, filters, cart, and wishlist, the user experience has been significantly improved. Debugging and performance optimization steps ensured a professional and scalable frontend.