

# ***Technical Report: Marketplace Development - Day 4***

**Naam::Muzna Amir**

**GIAIC Student: TUE 2-5PM**

**Time of Completion: 3:00 AM**

## **Introduction**

This report will outline the steps to build and integrate components for my marketplace, focusing on creating a dynamic product listing page and individual product detail pages. I will also discuss the challenges faced, the solutions implemented, and the best practices followed during development.

## Highlighted Code Snippets and Logic for Key Components and Integration

```
import Image from "next/image";
import Link from "next/link";
// =====
// React Icons
// =====
import { IoIosArrowForward } from "react-icons/io";

import { client } from "../../sanity/lib/client";
import { urlFor } from "../../sanity/lib/image";

export default async function Shop() {
  const Query = `*[_type == "product"] | order(id asc) {
    id,
    image,
    detail_btn,
    title,
    description,
    price,
    slug
  }`;

  const FetchData = await client.fetch(Query);
  console.log(FetchData);

  return (
    <div>
      <section>
        <div>
          <div>
            <div>
              <div className="flex flex-wrap py-32 px-10 relative mb-4">
                <Image
                  alt="shopcover_image"
                  className="block opacity-100 absolute inset-0 w-full h-full"
                  src="/images/shop_slide.png"
                  width = {600}
                  height = {300}
                />

                <div className="text-center relative z-10 w-full">
                  <h2 className="text-4xl md:text-6xl lg:text-6xl text-black font-bold font-sans mb-2">
                    Shop
                  </h2>
                  <button className="mt-3 text-black font-bold font-sans inline-flex items-center px-12 py-2 rounded-2xl">
                    <Link href = "/">Home</Link><IoIosArrowForward /><Link href = "/shop">Shop</Link>
                  </button>
                </div>
              </div>
            </div>
          </div>
        </div>
      </section>
    </div>
  );
}
```

```

"use client";
import { useState, useEffect } from "react";
import { client } from "../../../sanity/lib/client";
import { urlFor } from "../../../sanity/lib/image";
import Image from "next/image";
import Link from "next/link";

interface Props {
  id: number;
  image: string;
  title: string;
  price: string;
  description: string;
}

export default function ProductDetail({ params }) {
  const [quantity, setQuantity] = useState<number | null>(0);
  const query = `*[_type == "product" && slug.current == "${params.slug}"] {
    id,
    image,
    title,
    price,
    description
  }[0]`; // Fetch only one product

  const [product, setProduct] = useState<Props | null>(null);

  useEffect(() => {
    async function displayProduct() {
      try {
        const fetchedProduct = await client.fetch(query);
        setProduct(fetchedProduct);
      } catch (error) {
        console.error("Error fetching product:", error);
      }
    }
    displayProduct();
  }, [params.slug]);

  if (!product) {
    return <p>Loading or product not found...</p>;
  }

  const increment = () => setQuantity((prev) => (prev === null ? 1 : prev + 1));
  const decrement = () => setQuantity((prev) => (prev && prev > 1 ? prev - 1 : null));

```

```

import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';

// Load environment variables from .env.local
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname, '../.env.local') });

// Create Sanity client
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: false,
  token: process.env.SANITY_API_TOKEN,
  apiVersion: '2021-08-31',
});

async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);
    const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
    const buffer = Buffer.from(response.data);
    const asset = await client.assets.upload('image', buffer, {
      filename: imageUrl.split('/').pop(),
    });
    console.log(`Image uploaded successfully: ${asset._id}`);
    return asset._id;
  } catch (error) {
    console.error('Failed to upload image:', imageUrl, error);
    return null;
  }
}

async function deleteExistingProducts() {
  try {
    console.log('Deleting existing products from Sanity...');
    const existingProducts = await client.fetch('*[_type == "product"][_id]');
    for (const product of existingProducts) {
      await client.delete(product._id);
    }
    console.log('Existing products deleted successfully.');
```

## • Steps Taken to Build and Integrate Components

**Product Listing Page:** I created a product listing page that dynamically displays all products fetched from an API. The page uses React components to render product cards with relevant data such as title, image, price, etc.

**Individual Product Detail Pages:** I integrated individual product detail pages using dynamic routing. Each product has a unique ID, and based on that ID, the corresponding product data is fetched and displayed on a dedicated page.

## • Challenges Faced and Solutions Implemented

**Dynamic Routing:** While working on dynamic routing, I did not face major issues because I had already practiced dynamic routes in previous tasks. However, working on it in the context of the marketplace allowed me to fine-tune my understanding of routing in Next.js. I used the `useRouter` hook to successfully extract product IDs from the URL and display the relevant product details, making the routing process smooth.

**API Integration:** During the process of integrating the API for fetching product data, I didn't encounter significant errors as I had practiced handling API responses previously. The integration was seamless, and I used proper error handling to ensure the API returned the correct data before rendering. This approach enhanced my understanding of managing data fetching in real-world applications.

## • Best Practices Followed

**Modular Components:** I followed a modular approach by creating reusable components for product cards, which can be used both on the product listing page and the individual product detail pages.

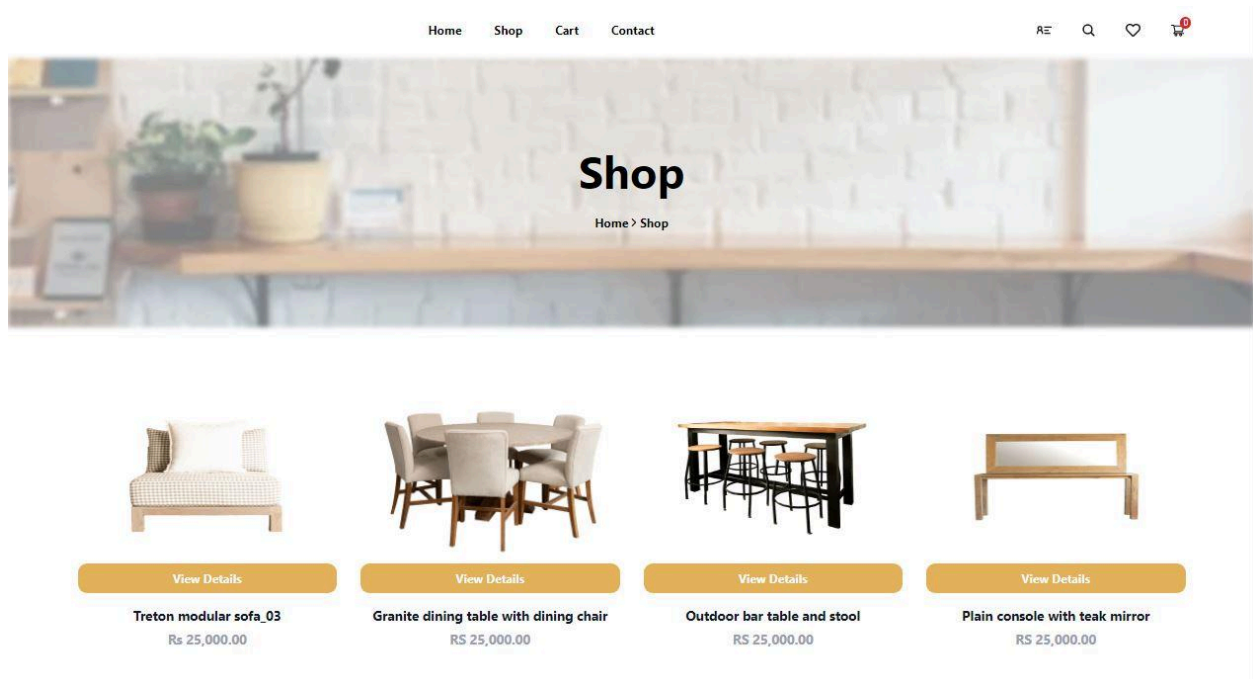
**Responsive Design:** I ensured that the pages were fully responsive and provided a consistent user experience across devices by utilizing CSS media queries and a mobile-first approach.

## Conclusion

This phase of the project focused on creating dynamic pages that render product data. The challenges encountered were successfully overcome by using effective solutions like dynamic routing and proper error handling. The best practices followed ensured a clean and maintainable codebase, as well as a responsive user interface.

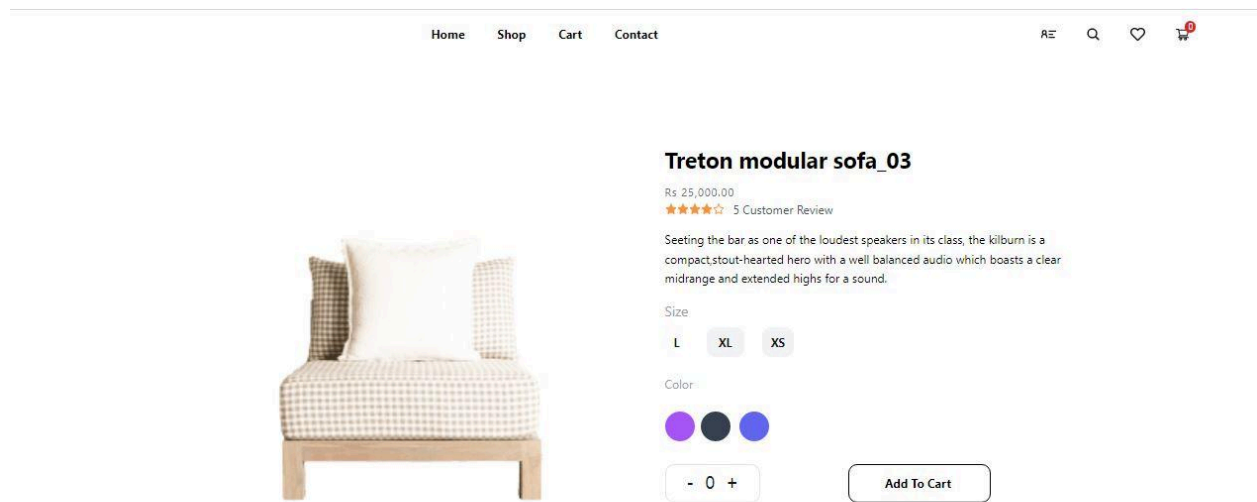
**Additional Insights:** I truly enjoy working through challenges like these, as they not only reinforce my learning but also keep the development process engaging. The opportunity to apply previously learned concepts in practical scenarios is always exciting, and I find it satisfying to solve issues as they arise.

## Dynamic Frontend Components for Product Listing, Detail Pages, and Pagination




This is a dynamic product listing page, where the data is seamlessly retrieved and displayed in real time from the API. The page is designed to automatically update as new product information is fetched, ensuring that users are presented with the most current and accurate

product details.



This page displays detailed information for each product, with accurate routing ensuring that the correct product data is rendered based on the user's selection. The dynamic content is fetched seamlessly, providing a smooth and informative user experience.






View Details

**Granite square side table**


RS 258,800.00



View Details

**Asgaard sofa dining chair**


RS 250,000.00



View Details

**Maya sofa three seater**

RS 115,000.00



View Details

**Outdoor sofa set**

RS 244,000.00

This section demonstrates the fully functional pagination feature, allowing users to navigate through multiple product pages with ease, enhancing the browsing experience by displaying data in manageable chunks.