Asia Parveen

00441760

Friday: 9am to 12pm

Mentor Sir Hamzah Syed

DAY 4 - BUILDING DYNAMIC FRONTEND COMPONENTS FOR MARKETPLACE:

Hackathon Task Description: Day 4

The task focuses on building dynamic frontend components for a marketplace application. Participants are required to create essential components that enhance user experience and interactivity.

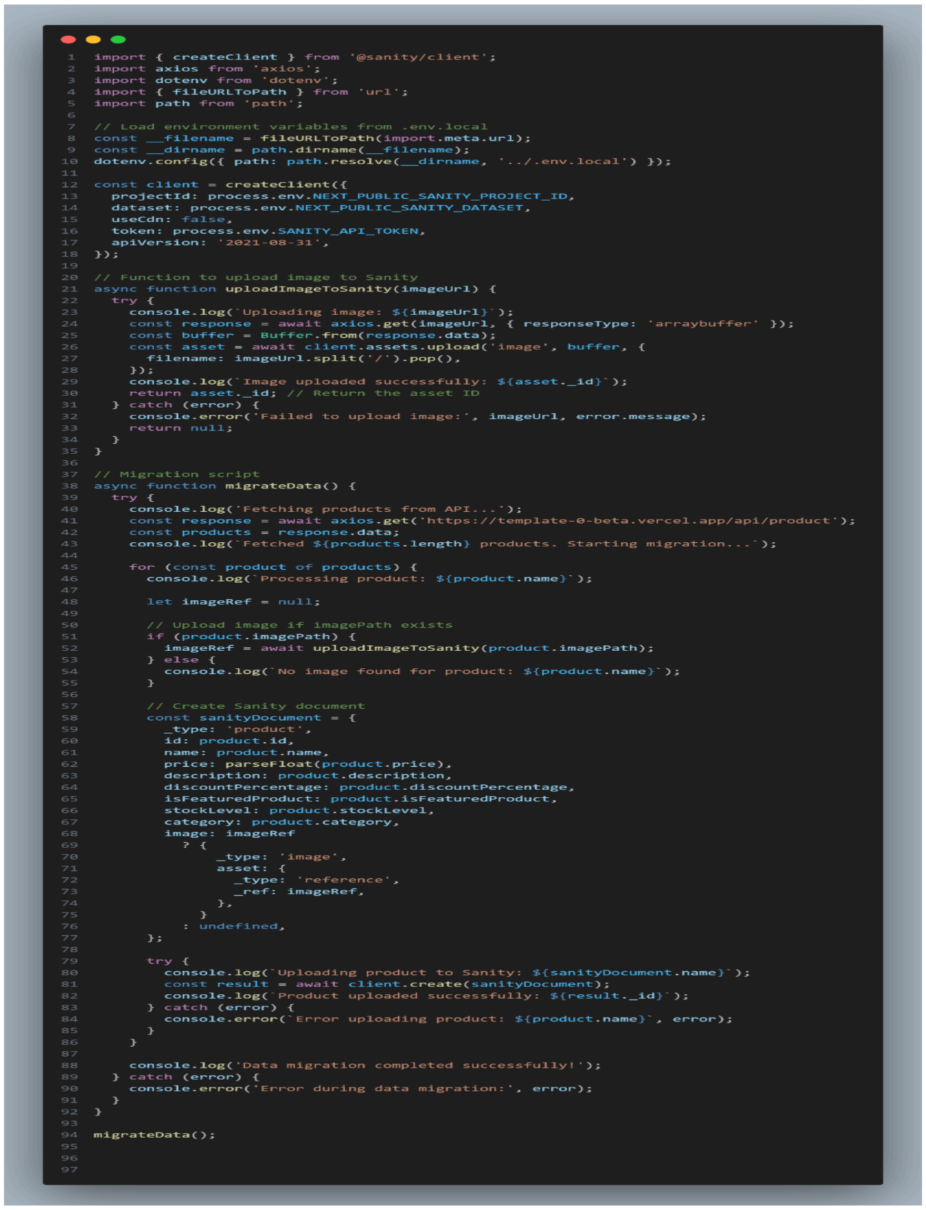
1. **Product Listing and Detail Components** to display products and their information dynamically.
2. **Category and Filter Panel Components** to enable users to browse and filter products efficiently.
3. **Search Bar** for quick product discovery.
4. **Cart, Wishlist, and Checkout Flow Components** to handle user actions and streamline the purchasing process.
5. **User Profile, Reviews, and Pagination Components** to manage user data and feedback while improving navigation.
6. **Footer, Header, and Notifications Components** to complete the layout and provide alerts for user actions.

This task emphasizes dynamic rendering, responsiveness, and user-centric design principles.

Migrating Data to Sanity

Sanity CMS was used to manage and organize the marketplace data efficiently. The process involved migrating product details, categories, user information, and other marketplace data into Sanity's schema. This step required creating a structured schema in Sanity Studio to define data types such as Products,Blogs, etc. After setting up the schema, data was imported or manually added into Sanity, ensuring the database was well-organized and ready for integration with the Next.js frontend.

Key Benefits of Using Sanity:

- Flexible content modeling.
- Real-time collaboration and updates.
- Scalable and highly structured data management.
- 

```
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') });

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',
});

// Function to upload image to Sanity
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; // Return the asset ID
  } catch (error) {
    console.error('Failed to upload image:', imageUrl, error.message);
    return null;
  }
}

// Migration script
async function migrateData() {
  try {
    console.log('Fetching products from API...');
    const response = await axios.get('https://template-0-beta.vercel.app/api/product');
    const products = response.data;
    console.log(`Fetched ${products.length} products. Starting migration...`);

    for (const product of products) {
      console.log(`Processing product: ${product.name}`);

      let imageRef = null;

      // Upload image if imagePath exists
      if (product.imagePath) {
        imageRef = await uploadImageToSanity(product.imagePath);
      } else {
        console.log(`No image found for product: ${product.name}`);
      }

      // Create Sanity document
      const sanityDocument = {
        _type: 'product',
        id: product.id,
        name: product.name,
        price: parseFloat(product.price),
        description: product.description,
        discountPercentage: product.discountPercentage,
        isFeaturedProduct: product.isFeaturedProduct,
        stockLevel: product.stockLevel,
        category: product.category,
        image: imageRef
          ? {
              _type: 'image',
              asset: {
                _type: 'reference',
                _ref: imageRef,
              },
            }
          : undefined,
      };

      try {
        console.log(`Uploading product to Sanity: ${sanityDocument.name}`);
        const result = await client.create(sanityDocument);
        console.log(`Product uploaded successfully: ${result._id}`);
      } catch (error) {
        console.error(`Error uploading product: ${product.name}`, error);
      }
    }

    console.log('Data migration completed successfully!');
  } catch (error) {
    console.error('Error during data migration:', error);
  }
}

migrateData();
```

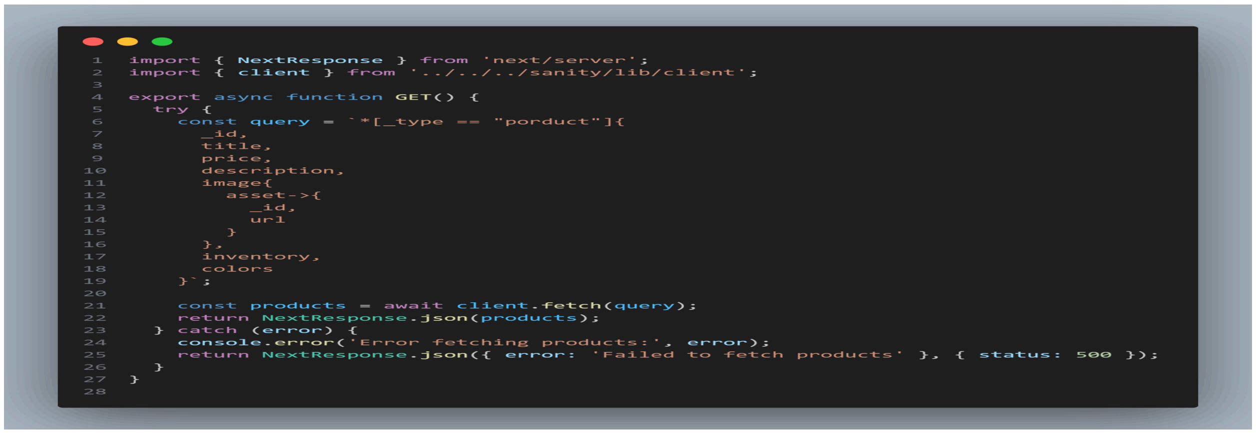
Fetching Data Using Queries in Next.js

After migrating data to Sanity, the frontend components in the Next.js application fetched this data using Sanity's GROQ (Graph-Relational Object Queries) or the official `@sanity/client` package. Queries were written to retrieve dynamic data like product listings, categories, and user reviews.

- Fetching a list of products based on a category.
- Retrieving product details like price, description, and images for the Product Detail Component.
- Fetching filtered or paginated data for the Search and Filter Panel.

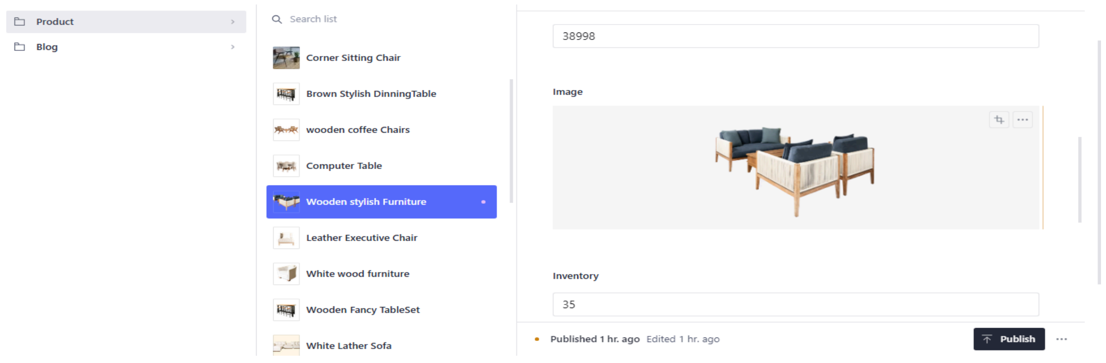
Integration Process:

1. Configure the Sanity client in the Next.js app with the project ID and dataset.
2. Write GROQ queries to fetch the required data.
3. Use Next.js `getStaticProps` or `getServerSideProps` for server-side data fetching, ensuring SEO optimization and dynamic rendering.



```
import { NextResponse } from 'next/server';
import { client } from '../../../sanity/lib/client';

export async function GET() {
  try {
    const query = `*[_type == "porduct"]{
      _id,
      title,
      price,
      description,
      image{
        asset->{
          _id,
          url
        }
      },
      inventory,
      colors
    }`;

    const products = await client.fetch(query);
    return NextResponse.json(products);
  } catch (error) {
    console.error('Error fetching products:', error);
    return NextResponse.json({ error: 'Failed to fetch products' }, { status: 500 });
  }
}

```

Import in sanity:



38998

Inventory

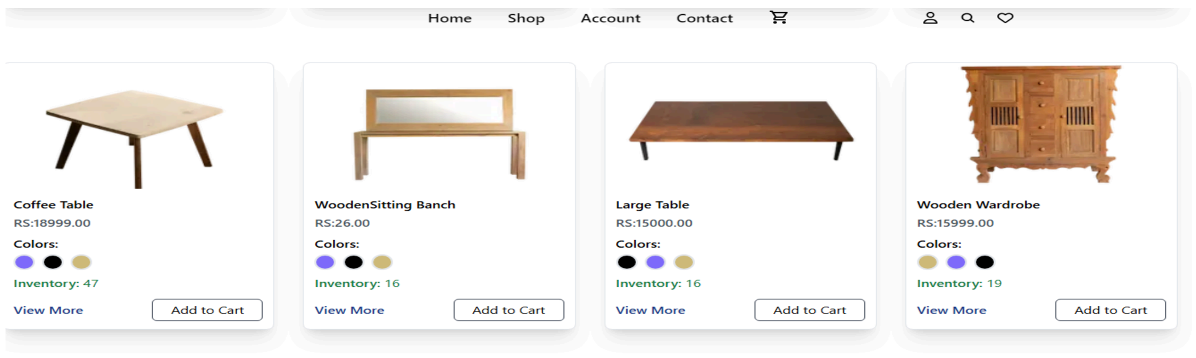
35

Published 1 hr. ago Edited 1 hr. ago

Product Listing with Sanity Integration

The **Product Listing Component** dynamically fetches product data from Sanity CMS, enabling efficient and real-time updates. Sanity serves as the backend, storing essential product details such as name, price, description, images, categories, and availability status.

1. **Data Structure in Sanity**:
   - Each product is defined with fields like `title`, `price`, `image`,stock, and `description`.
   - Categories and relationships are set up for filtering and organizing products.
2. **Fetching Products in Next.js**:
   - Using Sanity's **GROQ queries** and the `@sanity/client` package, the Next.js app retrieves product data.
   -



Coffee Table
RS:18999.00
Colors:
● ● ●
Inventory: 47

WoodenSitting Banch
RS:26.00
Colors:
● ● ●
Inventory: 16

Large Table
RS:15000.00
Colors:
● ● ●
Inventory: 16

Wooden Wardrobe
RS:15999.00
Colors:
● ● ●
Inventory: 19

3.

Dynamic Rendering:

- The component displays each product's details dynamically, including images, names, and prices, using the fetched data.
- Pagination or infinite scrolling is implemented for better navigation across large datasets.

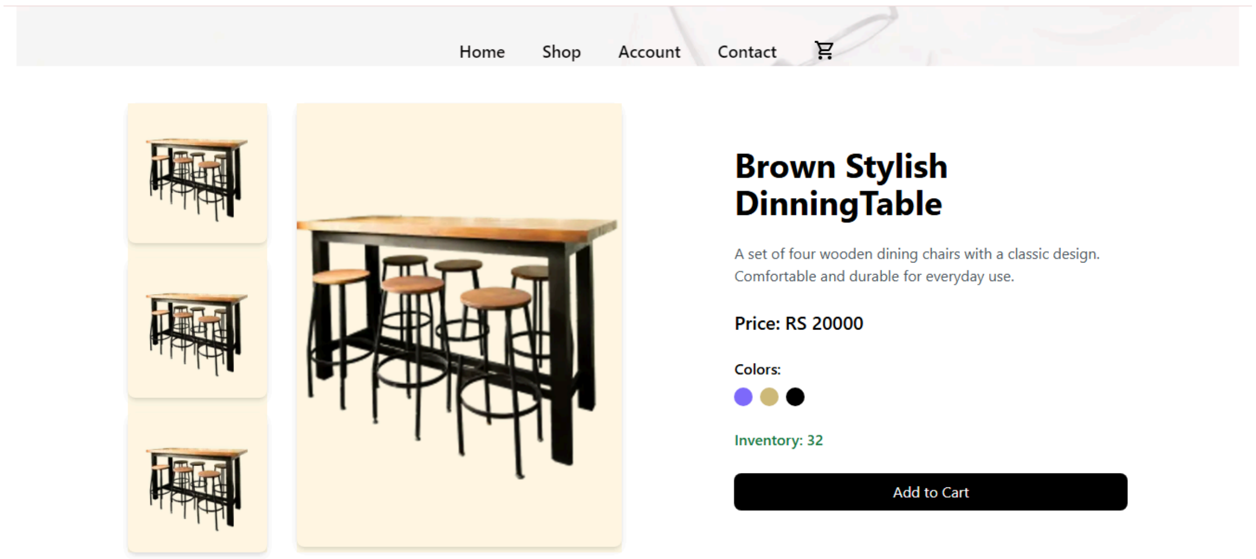
Product Detail Page with Dynamic Data

The **Product Detail Page** dynamically fetches and displays detailed information about a single product from Sanity CMS. This ensures that each product's unique attributes, such as descriptions, images, and specifications, are accurately rendered in real-time, providing users with an interactive and informative shopping experience.

Sanity Schema for Product Details

- Each product in Sanity has detailed fields like:
  - `title` (product name)
  - `price` (cost of the product)
  - `description` (detailed overview)
  - `images` (gallery of product photos)
  - `Stock` (product stock)
  - Colors

Dynamic Routing in Next.js

- Next.js uses dynamic routes (e.g., `/products/[slug]`) to generate individual product detail pages.
- 

Brown Stylish DinningTable

A set of four wooden dining chairs with a classic design. Comfortable and durable for everyday use.

Price: RS 20000

Colors:

Inventory: 32

-

Rendering Product Details

- The component displays:
  - **Product Title**: The name of the product.
  - **Image Gallery**: High-quality product images for better visualization.
  - **Price and Add to Cart Button**: Interactive options for purchase.
  - **Detailed Description**: A comprehensive overview of the product.
  - **Specifications Section**: Additional product information like materials, dimensions, or features.
  -

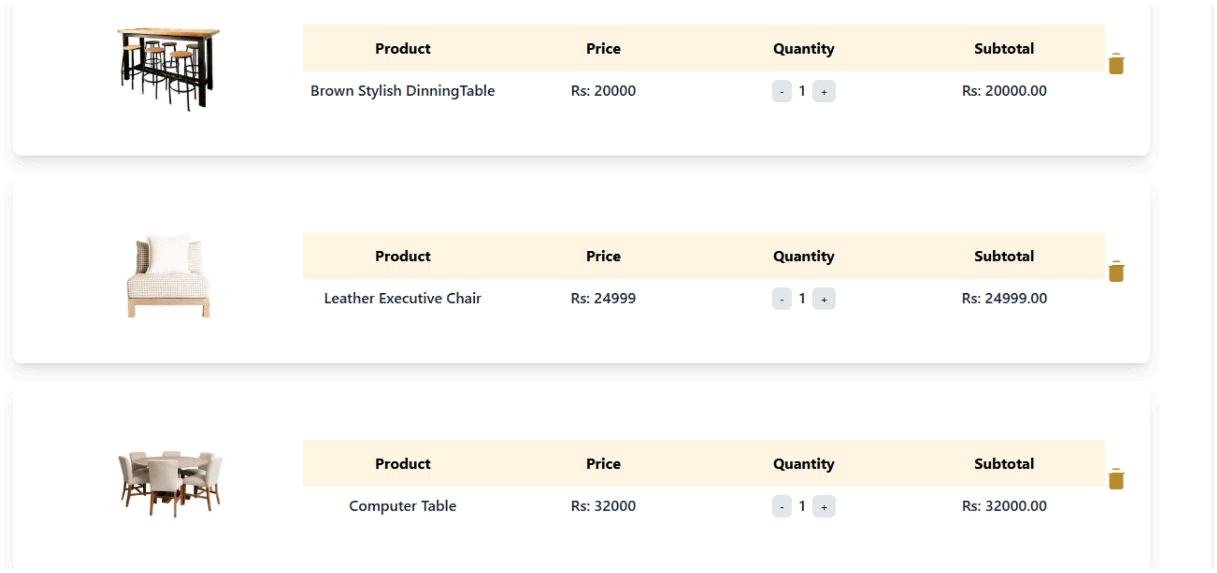
Cart Page Functionality

The **Cart Page** is designed to manage selected products for purchase, providing users with a seamless shopping experience. It dynamically updates as users add or remove items, ensuring real-time interaction.

1. **Add to Cart**: Products selected from the listing or detail page are added to the cart with their name, price, and quantity.
2. **Update Quantity**: Users can adjust the quantity of each item directly on the cart page, with the total price updating automatically.
3. **Remove Items**: Individual products can be removed from the cart with a single click.
4. **Total Price Calculation**: Displays a real-time total cost of all items in the cart.

5. **Proceed to Checkout**: Provides a button to navigate to the checkout page for completing the purchase.

The cart functionality ensures a user-friendly and efficient shopping workflow by managing product data dynamically in sync with the application state.



Product	Price	Quantity	Subtotal
Brown Stylish DinningTable	Rs: 20000	- 1 +	Rs: 20000.00

Product	Price	Quantity	Subtotal
Leather Executive Chair	Rs: 24999	- 1 +	Rs: 24999.00

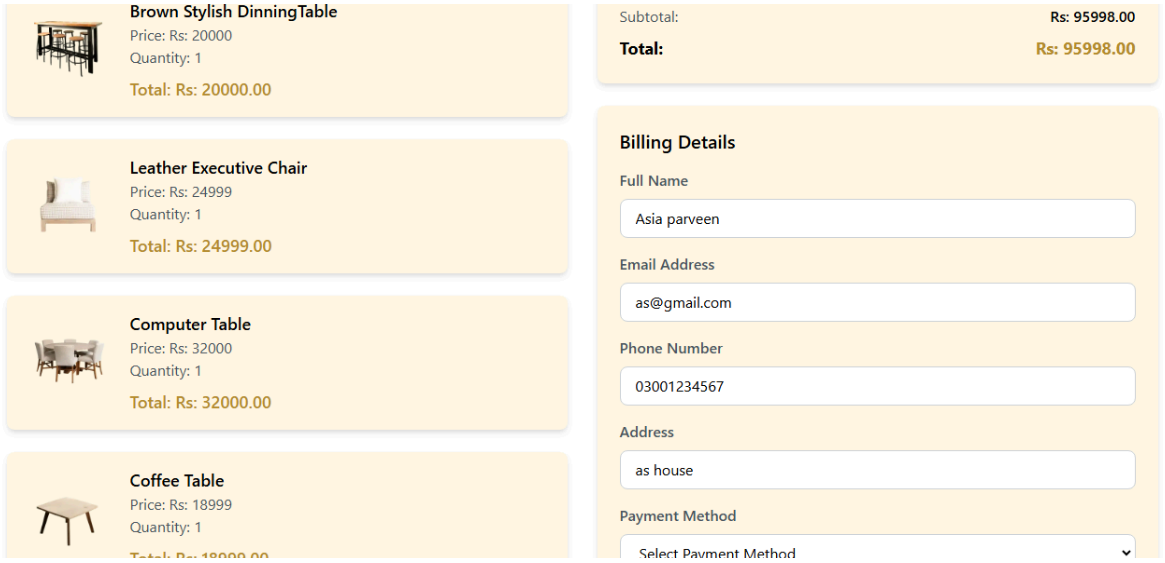
Product	Price	Quantity	Subtotal
Computer Table	Rs: 32000	- 1 +	Rs: 32000.00

Checkout Page Functionality

The **Checkout Page** is the final step in the purchasing process, allowing users to review their order and provide necessary details for payment and delivery. It ensures a smooth and secure transaction experience.

1. **Order Summary**:
   - Displays a detailed list of items in the cart, including product names, quantities, prices, and the total amount.
2. **User Information Form**:
   - Collects essential details such as name, address, phone number, and email for delivery purposes.
3. **Payment Options**:
   - Provides various secure payment methods (e.g., credit/debit card, PayPal, etc.) for completing the transaction.
4. **Promo Code Application**:
   - Allows users to apply discount codes for additional savings.
5. **Order Confirmation**:
   - After successful payment, displays an order confirmation message with an order ID and estimated delivery date.

The Checkout Page ensures a secure, user-friendly experience, guiding customers through the final steps of their purchase efficiently.


Brown Stylish DinningTable
Price: Rs: 20000
Quantity: 1
Total: Rs: 20000.00

Leather Executive Chair
Price: Rs: 24999
Quantity: 1
Total: Rs: 24999.00

Computer Table
Price: Rs: 32000
Quantity: 1
Total: Rs: 32000.00

Coffee Table
Price: Rs: 18999
Quantity: 1

Subtotal:	Rs: 95998.00
Total:	Rs: 95998.00

Billing Details

Full Name	Asia parveen
Email Address	as@gmail.com
Phone Number	03001234567
Address	as house
Payment Method	Select Payment Method

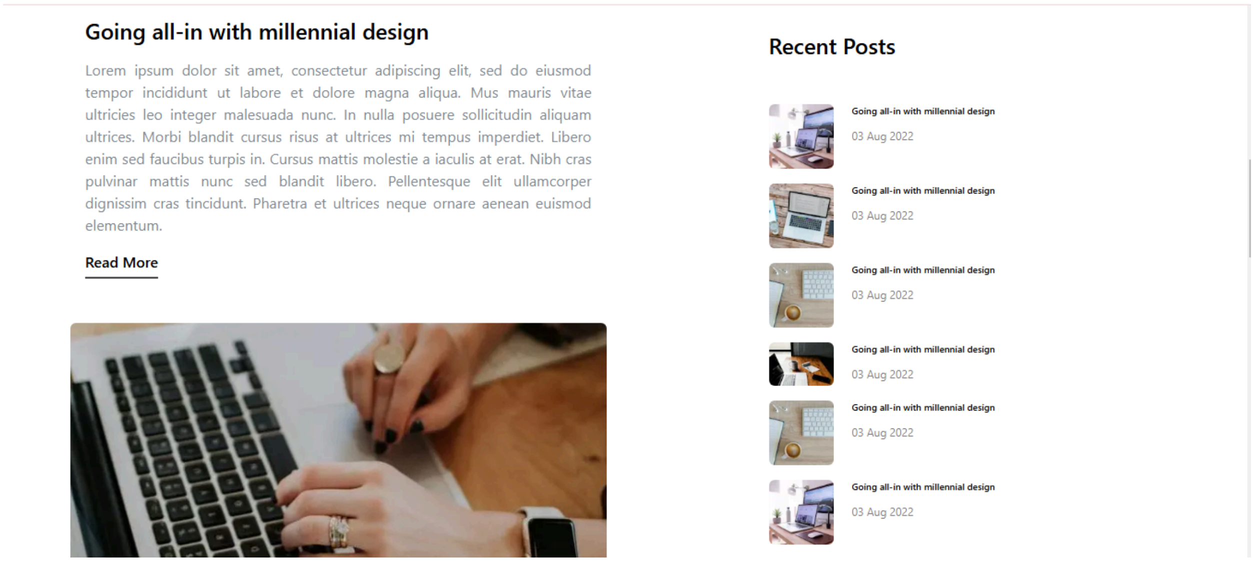
Blog Section with Dynamic Data from Sanity

The **Blog Section** dynamically fetches and displays blog posts from Sanity CMS, offering users valuable content such as updates, tutorials, or product-related articles.

1. **Sanity Data Structure**:
   - The blog posts are stored in Sanity with fields such as `title`, `slug`, `content`, `author`, `date`, and `image`.

Dynamic Blog Pages:

- Using **Next.js dynamic routing**, individual blog posts are displayed on separate pages, allowing users to read full articles.
  -



Going all-in with millennial design

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Mus mauris vitae ultricies leo integer malesuada nunc. In nulla posuere sollicitudin aliquam ultrices. Morbi blandit cursus risus at ultrices mi tempus imperdiet. Libero enim sed faucibus turpis in. Cursus mattis molestie a iaculis at erat. Nibh cras pulvinar mattis nunc sed blandit libero. Pellentesque elit ullamcorper dignissim cras tincidunt. Pharetra et ultrices neque ornare aenean euismod elementum.

Read More

Recent Posts

Going all-in with millennial design
03 Aug 2022

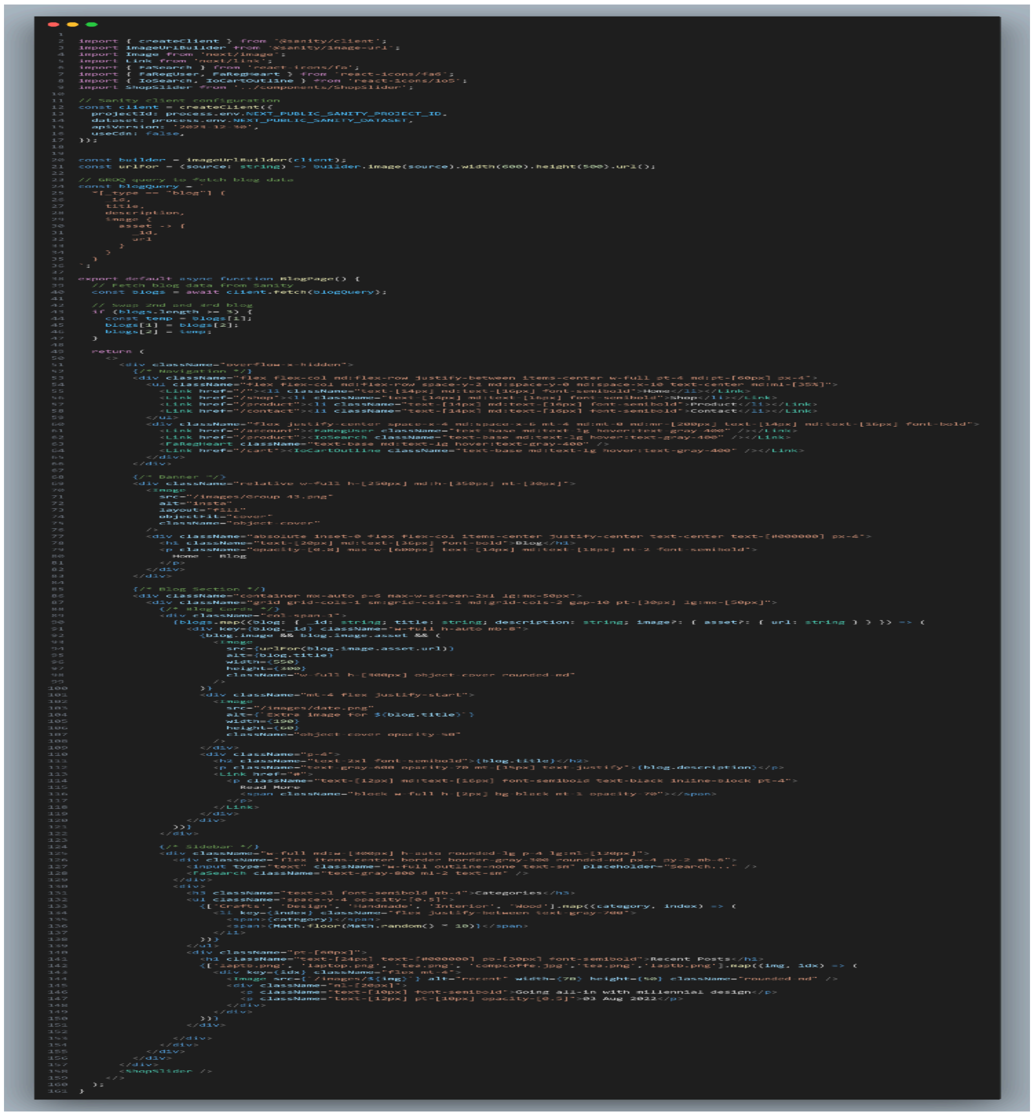
Going all-in with millennial design
03 Aug 2022

Going all-in with millennial design
03 Aug 2022

Going all-in with millennial design
03 Aug 2022

Going all-in with millennial design
03 Aug 2022

Going all-in with millennial design
03 Aug 2022



```tsx

import { createClient } from '@sanity/client';
import imageUrlBuilder from '@sanity/image-url';
import Image from 'next/image';
import Link from 'next/link';
import { FaSearch } from 'react-icons/fa';
import { FaRegUser, FaRegHeart } from 'react-icons/fa6';
import { IoSearch, IoCartOutline } from 'react-icons/io5';
import ShopSlider from '../components/ShopSlider';

// Sanity client configuration
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  apiVersion: '2023-12-30',
  useCdn: false,
});


const builder = imageUrlBuilder(client);
const urlFor = (source: string) => builder.image(source).width(600).height(500).url();

// GROQ query to fetch blog data
const blogQuery = `
  *[_type == "blog"] {
    _id,
    title,
    description,
    image {
      asset -> {
        _id,
        url
      }
    }
  }
`;

export default async function BlogPage() {
  // Fetch blog data from Sanity
  const blogs = await client.fetch(blogQuery);

  // Swap 2nd and 3rd blog
  if (blogs.length >= 3) {
    const temp = blogs[1];
    blogs[1] = blogs[2];
    blogs[2] = temp;
  }

  return (
    <>
      <div className="overflow-x-hidden">
        {/* Navigation */}
        <div className="flex flex-col md:flex-row justify-between items-center w-full pt-4 md:pt-[60px] px-4">
          <ul className="flex flex-col md:flex-row space-y-2 md:space-y-0 md:space-x-10 text-center md:ml-[35%]">
            <Link href="/"><li className="text-[14px] md:text-[16px] font-semibold">Home</li></Link>
            <Link href="/shop"><li className="text-[14px] md:text-[16px] font-semibold">Shop</li></Link>
            <Link href="/product"><li className="text-[14px] md:text-[16px] font-semibold">Product</li></Link>
            <Link href="/contact"><li className="text-[14px] md:text-[16px] font-semibold">Contact</li></Link>
          </ul>
          <div className="flex justify-center space-x-4 md:space-x-6 mt-4 md:mt-0 md:mr-[200px] text-[14px] md:text-[16px] font-bold">
            <Link href="/account"><FaRegUser className="text-base md:text-lg hover:text-gray-400" /></Link>
            <Link href="/product"><IoSearch className="text-base md:text-lg hover:text-gray-400" /></Link>
            <FaRegHeart className="text-base md:text-lg hover:text-gray-400" />
            <Link href="/cart"><IoCartOutline className="text-base md:text-lg hover:text-gray-400" /></Link>
          </div>
        </div>

        {/* Banner */}
        <div className="relative w-full h-[250px] md:h-[350px] mt-[30px]">
          <Image
            src="/images/Group 43.png"
            alt="insta"
            layout="fill"
            objectFit="cover"
            className="object-cover"
          />
          <div className="absolute inset-0 flex flex-col items-center justify-center text-center text-[#000000] px-4">
            <h1 className="text-[20px] md:text-[36px] font-bold">Blog</h1>
            <p className="opacity-[0.8] max-w-[600px] text-[14px] md:text-[18px] mt-2 font-semibold">
              Home - Blog
            </p>
          </div>
        </div>

        {/* Blog Section */}
        <div className="container mx-auto p-6 max-w-screen-2xl lg:mx-50px">
          <div className="grid grid-cols-1 sm:grid-cols-1 md:grid-cols-2 gap-10 pt-[30px] lg:mx-[50px]">
            {/* Blog Cards */}
            <div className="col-span-1">
              {blogs.map((blog: { _id: string; title: string; description: string; image?: { asset?: { url: string } } }) => (
                <div key={blog._id} className="w-full h-auto mb-8">
                  {blog.image && blog.image.asset && (
                    <Image
                      src={urlFor(blog.image.asset.url)}
                      alt={blog.title}
                      width={550}
                      height={300}
                      className="w-full h-[300px] object-cover rounded-md"
                    />
                  )}
                  <div className="mt-4 flex justify-start">
                    <Image
                      src="/images/date.png"
                      alt={`Extra image for ${blog.title}`}
                      width={190}
                      height={60}
                      className="object-cover opacity-50"
                    />
                  </div>
                  <div className="p-4">
                    <h2 className="text-2xl font-semibold">{blog.title}</h2>
                    <p className="text-gray-600 opacity-70 mt-[15px] text-justify">{blog.description}</p>
                    <Link href="#">
                      <p className="text-[12px] md:text-[16px] font-semibold text-black inline-block pt-4">
                        Read More
                        <span className="block w-full h-[2px] bg-black mt-1 opacity-70"></span>
                      </p>
                    </Link>
                  </div>
                </div>
              ))}
            </div>

            {/* Sidebar */}
            <div className="w-full md:w-[300px] h-auto rounded-lg p-4 lg:ml-[120px]">
              <div className="flex items-center border border-gray-300 rounded-md px-4 py-2 mb-6">
                <input type="text" className="w-full outline-none text-sm" placeholder="Search..." />
                <FaSearch className="text-gray-800 ml-2 text-sm" />
              </div>
              <div>
                <h3 className="text-xl font-semibold mb-4">Categories</h3>
                <ul className="space-y-4 opacity-[0.5]">
                  {['Crafts', 'Design', 'Handmade', 'Interior', 'Wood'].map((category, index) => (
                    <li key={index} className="flex justify-between text-gray-700">
                      <span>{category}</span>
                      <span>{Math.floor(Math.random() * 10)}</span>
                    </li>
                  ))}
                </ul>
                <div className="pt-[60px]">
                  <h1 className="text-[24px] text-[#000000] pb-[30px] font-semibold">Recent Posts</h1>
                  {['laptb.png', 'laptop.png', 'tea.png', 'compcoffe.jpg','tea.png','laptb.png'].map((img, idx) => (
                    <div key={idx} className="flex mt-4">
                      <Image src={`/images/${img}`} alt="recent" width={70} height={50} className="rounded-md" />
                      <div className="ml-[20px]">
                        <p className="text-[10px] font-semibold">Going all-in with millennial design</p>
                        <p className="text-[12px] pt-[10px] opacity-[0.5]">03 Aug 2022</p>
                      </div>
                    </div>
                  ))}
                </div>

              </div>
            </div>
          </div>
        </div>
      </div>
      <ShopSlider />
    </>
  );
}
```

Customer Reviews Section with Dynamic Data

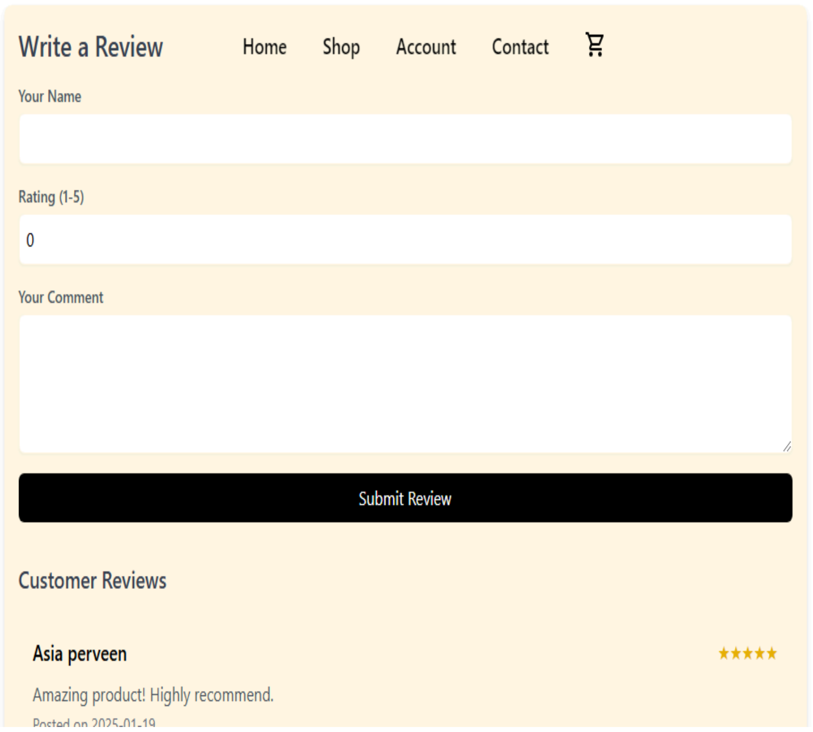
The **Customer Reviews Section** enables users to read and submit product feedback, helping future buyers make informed decisions. This section pulls review data from Sanity, keeping it up-to-date and easy to manage.

- Store reviews in real time in the bottom or the reviews section

2. **Displaying Reviews**:
    - Reviews are shown on the product detail page with the user's name, rating, and comments.
    - A review submission form can also be provided, allowing customers to share their experiences after purchasing a product.

The integration of dynamic content from Sanity for both blog posts and customer reviews ensures that the website remains fresh and engaging while offering valuable information to users.

3.



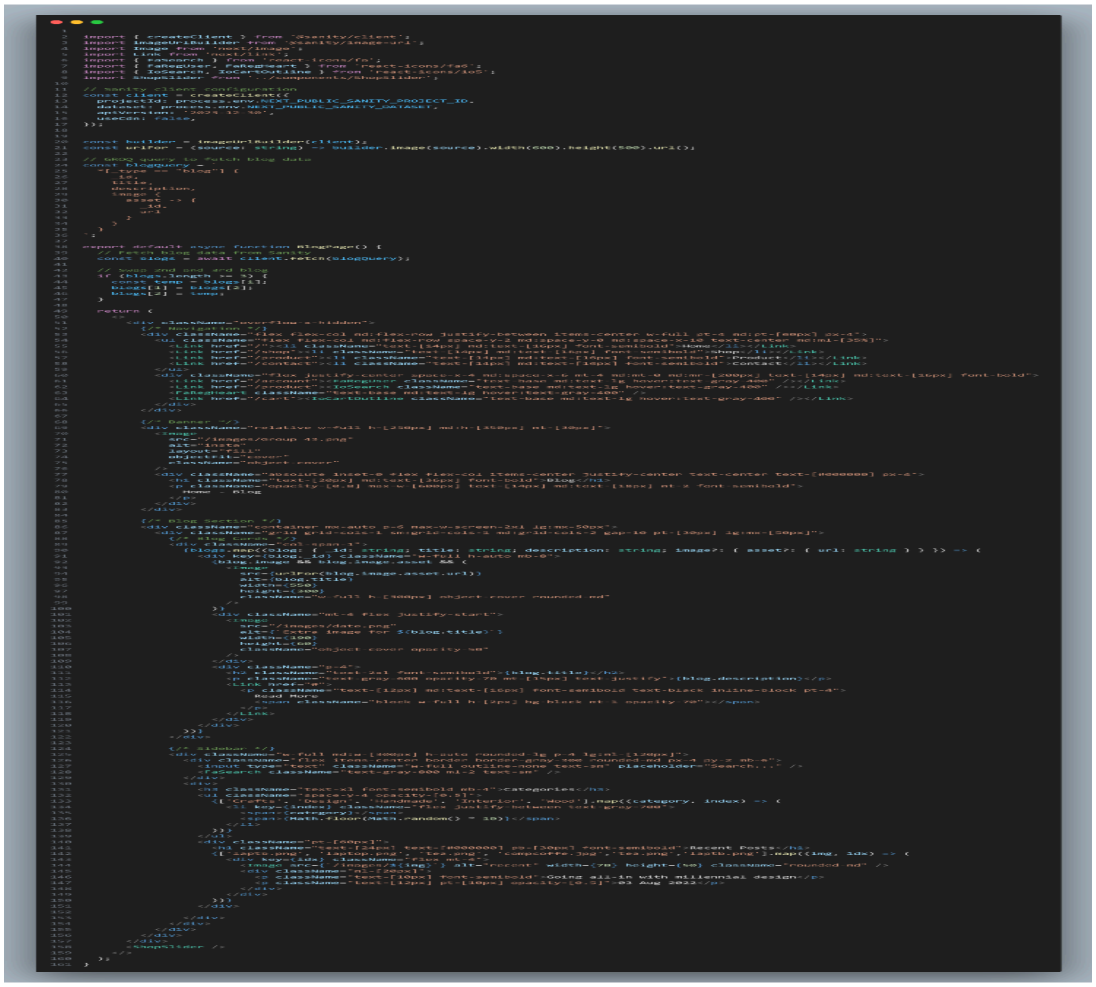
0

Customer Reviews

Asia perveen ★★★★★

Amazing product! Highly recommend.

Posted on 2025-01-19



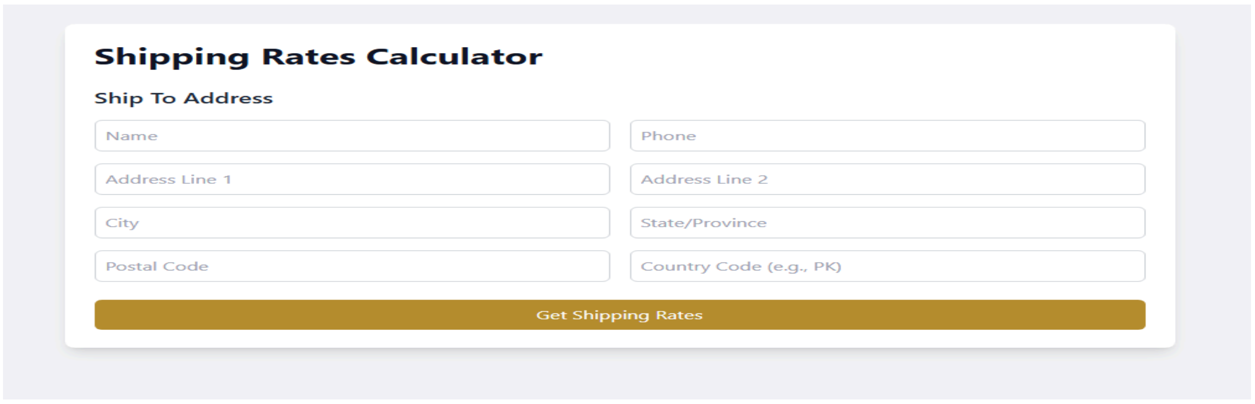
```
import { createClient } from '@sanity/client';
import imageUrlBuilder from '@sanity/image-url';
import Image from 'next/image';
import Link from 'next/link';
import { FaSearch } from 'react-icons/fa';
import { FaRegUser, FaRegHeart } from 'react-icons/fa6';
import { IoSearch, IoCartOutline } from 'react-icons/io5';
import ShopSlider from '../components/ShopSlider';

// Sanity client configuration
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  apiVersion: '2023-12-30',
  useCdn: false,
});


const builder = imageUrlBuilder(client);
const urlFor = (source: string) => builder.image(source).width(600).height(500).url();

// GROQ query to fetch blog data
const blogQuery = `
  *[_type == "blog"] {
    _id,
    title,
    description,
    image {
      asset -> {
        _id,
        url
      }
    }
  }
`;

export default async function BlogPage() {
  // Fetch blog data from Sanity
  const blogs = await client.fetch(blogQuery);

  // Swap 2nd and 3rd blog
  if (blogs.length >= 3) {
    const temp = blogs[1];
    blogs[1] = blogs[2];
    blogs[2] = temp;
  }

  return (
    <>
      <div className="overflow-x-hidden">
        {/* Navigation */}
        <div className="flex flex-col md:flex-row justify-between items-center w-full pt-4 md:pt-[60px] px-4">
          <ul className="flex flex-col md:flex-row space-y-2 md:space-y-0 md:space-x-10 text-center md:ml-[35%]">
            <Link href="/"><li className="text-[14px] md:text-[16px] font-semibold">Home</li></Link>
            <Link href="/shop"><li className="text-[14px] md:text-[16px] font-semibold">Shop</li></Link>
            <Link href="/product"><li className="text-[14px] md:text-[16px] font-semibold">Product</li></Link>
            <Link href="/contact"><li className="text-[14px] md:text-[16px] font-semibold">Contact</li></Link>
          </ul>
          <div className="flex justify-center space-x-4 md:space-x-6 mt-4 md:mt-0 md:mr-[200px] text-[14px] md:text-[16px] font-bold">
            <Link href="/account"><FaRegUser className="text-base md:text-lg hover:text-gray-400" /></Link>
            <Link href="/product"><IoSearch className="text-base md:text-lg hover:text-gray-400" /></Link>
            <FaRegHeart className="text-base md:text-lg hover:text-gray-400" />
            <Link href="/cart"><IoCartOutline className="text-base md:text-lg hover:text-gray-400" /></Link>
          </div>
        </div>

        {/* Banner */}
        <div className="relative w-full h-[250px] md:h-[350px] mt-[30px]">
          <Image
            src="/images/Group 43.png"
            alt="insta"
            layout="fill"
            objectFit="cover"
            className="object-cover"
          />
          <div className="absolute inset-0 flex flex-col items-center justify-center text-center text-[#000000] px-4">
            <h1 className="text-[20px] md:text-[36px] font-bold">Blog</h1>
            <p className="opacity-[0.8] max-w-[600px] text-[14px] md:text-[18px] mt-2 font-semibold">
              Home - Blog
            </p>
          </div>
        </div>

        {/* Blog Section */}
        <div className="container mx-auto p-6 max-w-screen-2xl lg:mx-50px">
          <div className="grid grid-cols-1 sm:grid-cols-1 md:grid-cols-2 gap-10 pt-[30px] lg:mx-[50px]">
            {/* Blog Cards */}
            <div className="col-span-1">
              {blogs.map((blog: { _id: string; title: string; description: string; image?: { asset?: { url: string } } }) => (
                <div key={blog._id} className="w-full h-auto mb-8">
                  {blog.image && blog.image.asset && (
                    <Image
                      src={urlFor(blog.image.asset.url)}
                      alt={blog.title}
                      width={550}
                      height={300}
                      className="w-full h-[300px] object-cover rounded-md"
                    />
                  )}
                  <div className="mt-4 flex justify-start">
                    <Image
                      src="/images/date.png"
                      alt={`Extra image for ${blog.title}`}
                      width={190}
                      height={60}
                      className="object-cover opacity-50"
                    />
                  </div>
                  <div className="p-4">
                    <h2 className="text-2xl font-semibold">{blog.title}</h2>
                    <p className="text-gray-600 opacity-70 mt-[15px] text-justify">{blog.description}</p>
                    <Link href="#">
                      <p className="text-[12px] md:text-[16px] font-semibold text-black inline-block pt-4">
                        Read More
                        <span className="block w-full h-[2px] bg-black mt-1 opacity-70"></span>
                      </p>
                    </Link>
                  </div>
                </div>
              ))}
            </div>

            {/* Sidebar */}
            <div className="w-full md:w-[400px] h-auto rounded-lg p-4 lg:ml-[120px]">
              <div className="flex items-center border border-gray-300 rounded-md px-4 py-2 mb-6">
                <input type="text" className="w-full outline-none text-sm" placeholder="Search..." />
                <FaSearch className="text-gray-800 ml-2 text-sm" />
              </div>
              <div>
                <h3 className="text-xl font-semibold mb-4">Categories</h3>
                <ul className="space-y-4 opacity-[0.5]">
                  {['Crafts', 'Design', 'Handmade', 'Interior', 'Wood'].map((category, index) => (
                    <li key={index} className="flex justify-between text-gray-700">
                      <span>{category}</span>
                      <span>{Math.floor(Math.random() * 10)}</span>
                    </li>
                  ))}
                </ul>
                <div className="pt-[60px]">
                  <h1 className="text-[24px] text-[#000000] pb-[30px] font-semibold">Recent Posts</h1>
                  {['laptb.png', 'laptop.png', 'tea.png', 'compcoffe.jpg','tea.png','laptb.png'].map((img, idx) => (
                    <div key={idx} className="flex mt-4">
                      <Image src={`/images/${img}`} alt="recent" width={70} height={50} className="rounded-md" />
                      <div className="ml-[20px]">
                        <p className="text-[10px] font-semibold">Going all-in with millennial design</p>
                        <p className="text-[12px] pt-[10px] opacity-[0.5]">03 Aug 2022</p>
                      </div>
                    </div>
                  ))}
                </div>

              </div>
            </div>
          </div>
        </div>
      </div>
      <ShopSlider />
    </>
  );
}
```

Shipment Functionality

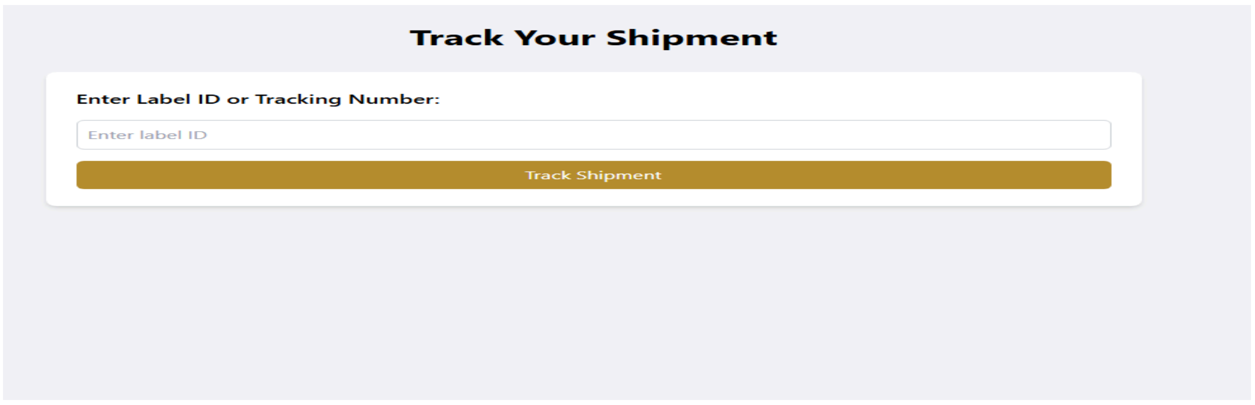
The **Shipment Section** is an essential part of the checkout and order fulfillment process, ensuring that customers' products are delivered accurately and efficiently. It integrates with a shipping service (like ShipEngine) to handle shipping calculations, tracking, and management of delivery options.

1. **integration with ShipEngine**:
    - ShipEngine is integrated into the system to provide real-time shipping rates, generate labels, and track shipments.
    - It uses data from the customer’s address and the product weight/dimensions to calculate the best available shipping options.
    - The integration also allows for accurate shipping costs based on the customer’s location and preferred delivery method (e.g., ground, express, or international shipping).
2. **Fetching Shipping Rates**:

    - During the checkout process, the user's shipping address is passed to the ShipEngine API to fetch the available shipping methods and their associated costs.
    - The user can then choose their preferred shipping method before proceeding to payment.
3. **Shipment Tracking**:
    - After the order is placed and the shipment is processed, customers can track their packages using tracking numbers generated by ShipEngine.
    - A tracking link is provided to the customer, allowing them to monitor the progress of their shipment in real-time.
4. **Shipment Status Updates**:
    - Customers receive notifications regarding their shipment status, including when it's dispatched, in transit, and delivered.
    - Any delays or issues are also communicated to customers to keep them informed throughout the delivery process.



Tracking:



I will make my Ecommerece website more efficient and add more features and components in further in sha Allah: Good by