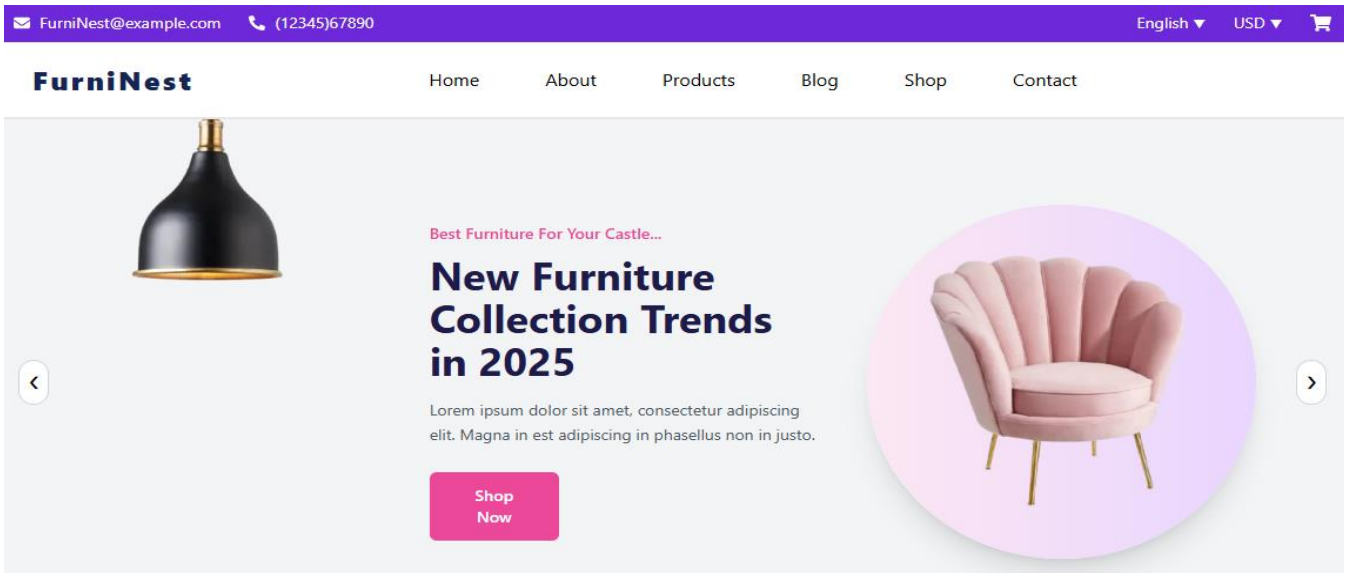

FurniNest@example.com
(12345)67890
English
USD
FurniNest
Home
About
Products
Blog
Shop
Contact
Best Furniture For Your Castle...
New Furniture Collection Trends in 2025
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Magna in est adipiscing in phasellus non in justo.
Shop Now

Day 4 - Dynamic Frontend Components

- FURNINEST

FATIMA AKIF | roll no: 00312734

Technical Report - FurniNest

Prepared by: Fatima Akif

FurniNest is a dynamic e-commerce marketplace specializing in furniture sales. It utilizes **Sanity, Stripe, and Clerk** as core technologies to ensure efficient product management, secure transactions, and user authentication. This report outlines the steps taken to integrate these components, the challenges encountered, and the best practices followed during development.

STEPS TAKEN TO BUILD AND INTEGRATE COMPONENTS

1. Product Management with Sanity CMS

- Configured **Sanity Studio** to define a structured schema for products, including fields for `title`, `description`, `price`, `category`, `images`, and `slug`.
- Implemented **dynamic slug-based routing** in **Next.js** to generate product detail pages dynamically.
- Used **Sanity's GROQ queries** to fetch product data based on the `slug`.

2. User Authentication with Clerk

- Integrated **Clerk** to handle user authentication.
- Configured **sign-up, sign-in, and session management** for seamless user experience.
- Implemented role-based access control (e.g., restricting admin features).

3. Secure Payment Processing with Stripe

- Configured **Stripe Checkout** for handling transactions.
- Implemented **webhooks** to listen for payment confirmations and update order statuses.
- Ensured proper error handling for failed payments.

CHALLENGES FACED AND SOLUTIONS IMPLEMENTED

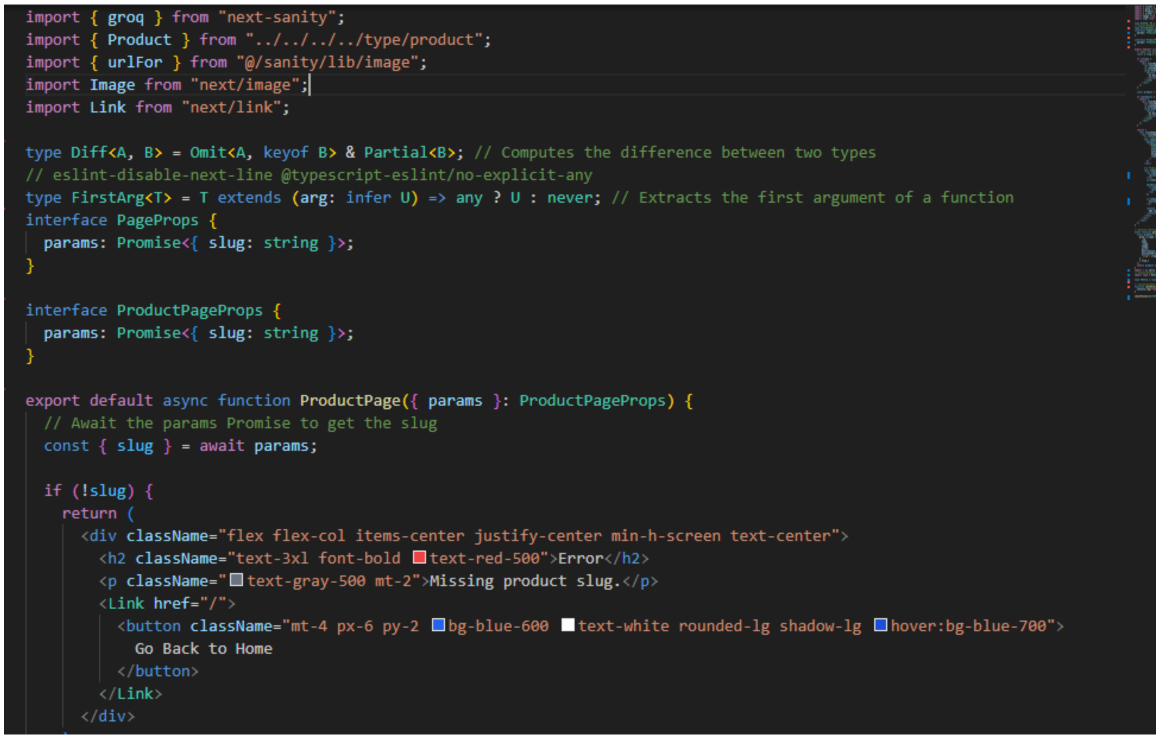
Challenge	Solution Implemented
Dynamic product page routing using `slug`	Used Next.js `getStaticPaths` and `getStaticProps` to pre-generate product pages.
Authentication state persistence	Used Clerk's session handling and Next.js middleware to maintain login state.
Handling out-of-stock products	Added a `stock` field in Sanity and implemented a check before allowing purchases.
Securely storing API keys	Used environment variables (`.env.local`) to protect sensitive credentials.
Real-time updates to products	Implemented **Sanity Webhooks** to automatically update products on the frontend when changes are made in Sanity CMS.

BEST PRACTICES FOLLOWED DURING DEVELOPMENT

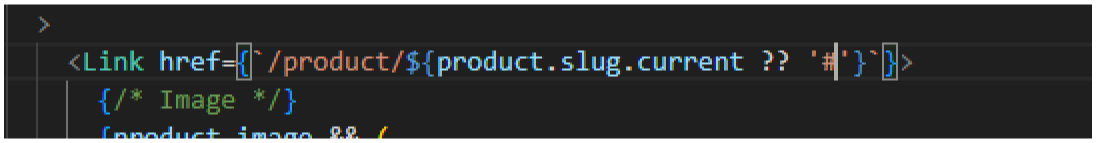
1. **Modular Code Structure**
   - Organized components into separate directories (`components`, `pages`, `lib`).
   - Used reusable UI components for product listings and details.
2. **Optimized API Calls**
   - Used **GROQ queries** to fetch only necessary data from Sanity, improving performance.
   - Implemented **server-side rendering** (SSR) **and static generation** (SSG) for efficient data fetching.
3. **Security Measures**
   - Stored sensitive API keys in `.env.local` and configured **Next.js API routes** for backend logic.
   - Used **Clerk's JWT-based authentication** to ensure secure user sessions.
4. **User Experience Enhancements**
   - Added a **loading state** for dynamic pages to improve performance.
   - Implemented **lazy loading** for product images using Next.js's `next/image`.

CONCLUSION

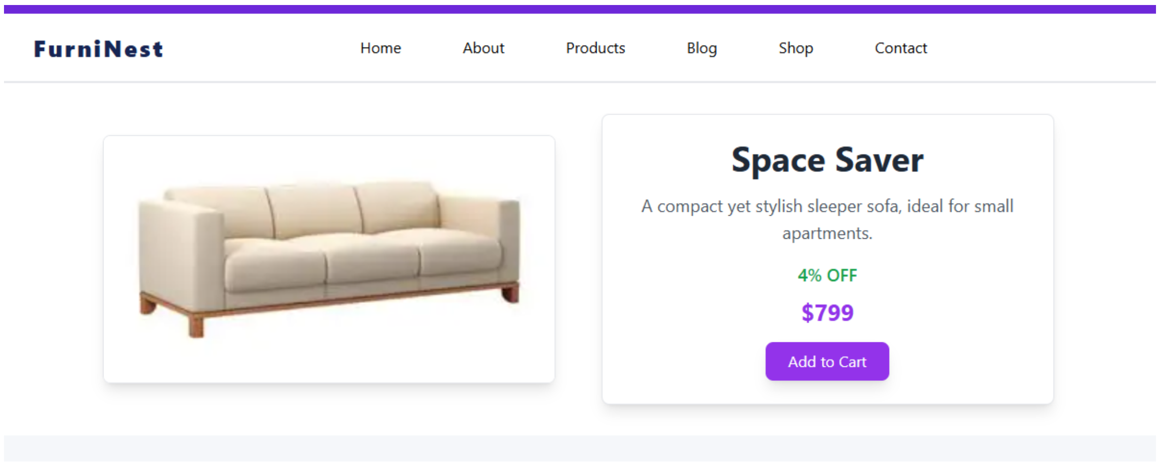
By integrating **Sanity CMS, Stripe, and Clerk**, FurniNest successfully delivers a seamless shopping experience with dynamic product pages, secure authentication, and efficient transactions. The challenges faced during development were addressed through structured solutions, and best practices were followed to maintain scalability, performance, and security.



```
import { groq } from "next-sanity";
import { Product } from "../../../../type/product";
import { urlFor } from "@/sanity/lib/image";
import Image from "next/image";
import Link from "next/link";

type Diff<A, B> = Omit<A, keyof B> & Partial<B>; // Computes the difference between two types
// eslint-disable-next-line @typescript-eslint/no-explicit-any
type FirstArg<T> = T extends (arg: infer U) => any ? U : never; // Extracts the first argument of a function
interface PageProps {
  params: Promise<{ slug: string }>;
}

interface ProductPageProps {
  params: Promise<{ slug: string }>;
}

export default async function ProductPage({ params }: ProductPageProps) {
  // Await the params Promise to get the slug
  const { slug } = await params;

  if (!slug) {
    return (
      <div className="flex flex-col items-center justify-center min-h-screen text-center">
        <h2 className="text-3xl font-bold text-red-500">Error</h2>
        <p className="text-gray-500 mt-2">Missing product slug.</p>
        <Link href="/">
          <button className="mt-4 px-6 py-2 bg-blue-600 text-white rounded-lg shadow-lg hover:bg-blue-700">
            Go Back to Home
          </button>
        </Link>
      </div>
```



```
  >
  <Link href={`/product/${product.slug.current ?? '#'}`}>
    {/* Image */}
    {product.image && (
```



Space Saver

A compact yet stylish sleeper sofa, ideal for small apartments.

4% OFF

$799