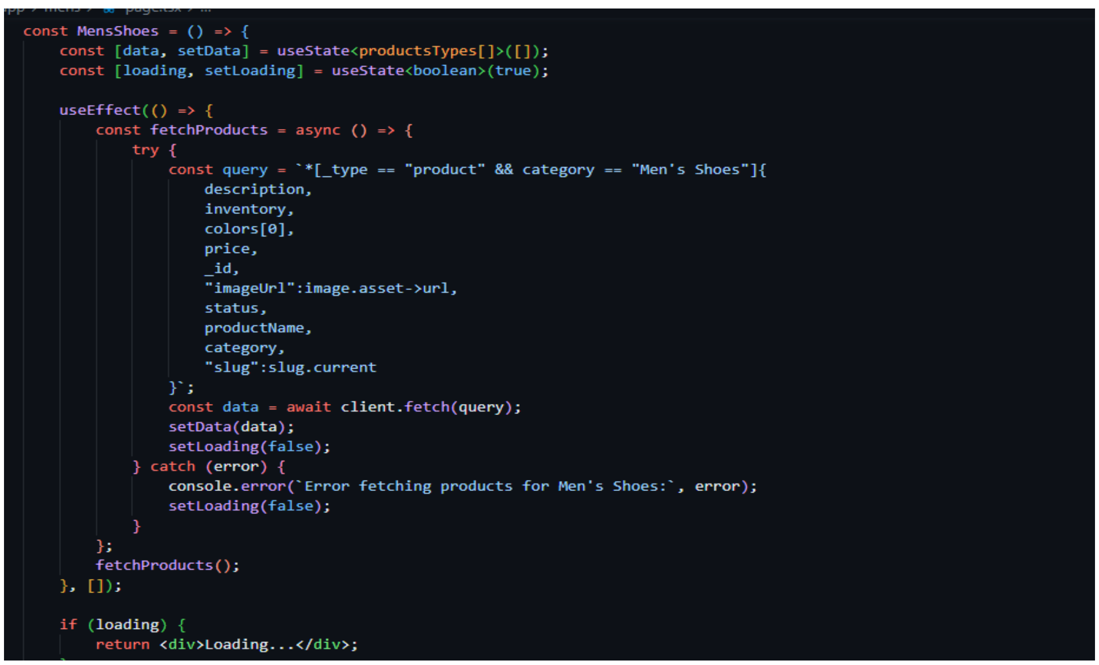
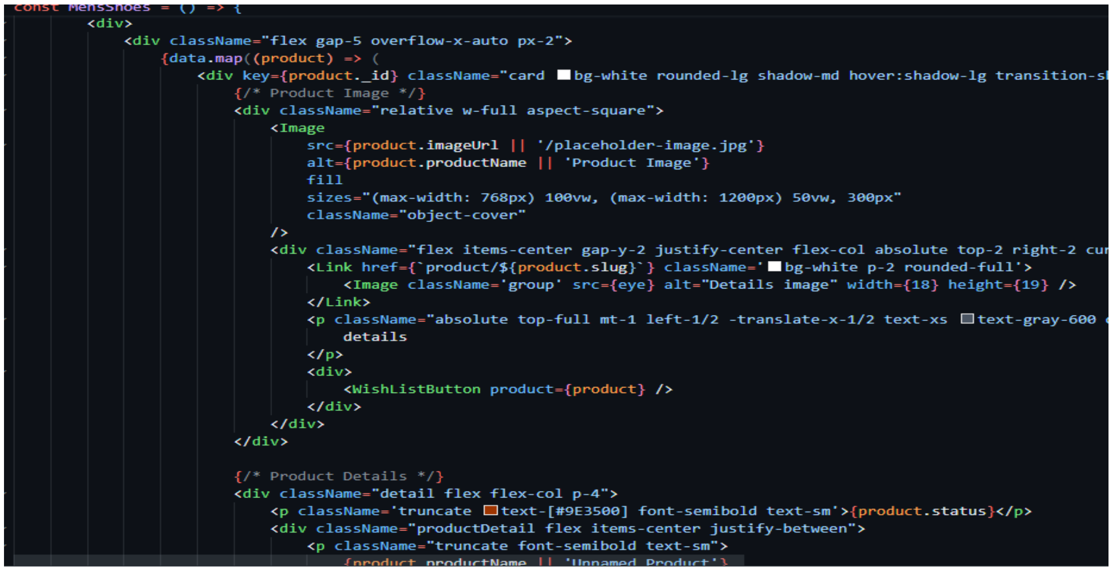
1. Custom Migration Code

- **Purpose:** Transfers data from **Sanity CMS** to the Nike Store **database**.
- **Steps Involved:**
    - Connects to **Sanity API** using authentication tokens.
    - Fetches structured content via **GROQ queries**.
    - Maps and reformats data to match the **schema**.
    - Saves data into the database using **REST API calls**.
    - Uses **bulk data insertion** for efficiency and **error handling** to prevent crashes.
    - 

```
const MensShoes = () => {
    const [data, setData] = useState<productsTypes[]>([]);
    const [loading, setLoading] = useState<boolean>(true);

    useEffect(() => {
        const fetchProducts = async () => {
            try {
                const query = `*[_type == "product" && category == "Men's Shoes"]{
                    description,
                    inventory,
                    colors[0],
                    price,
                    _id,
                    "imageUrl":image.asset->url,
                    status,
                    productName,
                    category,
                    "slug":slug.current
                }`;
                const data = await client.fetch(query);
                setData(data);
                setLoading(false);
            } catch (error) {
                console.error(`Error fetching products for Men's Shoes:`, error);
                setLoading(false);
            }
        };
        fetchProducts();
    }, []);

    if (loading) {
        return <div>Loading...</div>;
```

2. Client Page Code (Next.js Frontend)

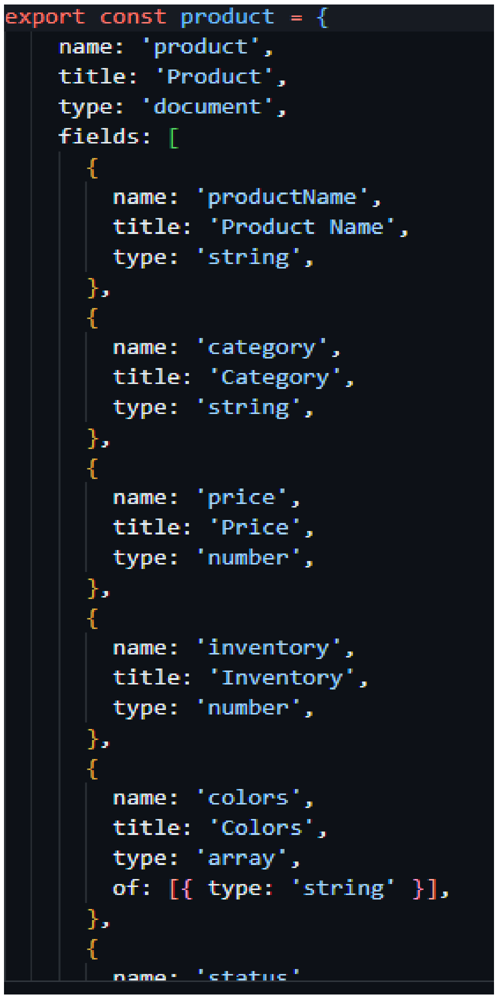
- **Data Fetching:** Uses `getServerSideProps` with **GROQ queries** for **server-side rendering (SSR)**.
- **Rendering:**
    - Displays **Shoes items** using React's `.map()` function.
    - Supports **dynamic routing** (`/product/[id]`) for detailed product pages.
- **Optimizations:**
    - **SEO-friendly** via SSR.
    - **Responsive design** for various screen sizes.

○ 

```
const MensShoes = () => {
    <div>
        <div className="flex gap-5 overflow-x-auto px-2">
            {data.map((product) => (
                <div key={product._id} className="card ⬜bg-white rounded-lg shadow-md hover:shadow-lg transition-s
                    {/* Product Image */}
                    <div className="relative w-full aspect-square">
                        <Image
                            src={product.imageUrl || '/placeholder-image.jpg'}
                            alt={product.productName || 'Product Image'}
                            fill
                            sizes="(max-width: 768px) 100vw, (max-width: 1200px) 50vw, 300px"
                            className="object-cover"
                        />
                        <div className="flex items-center gap-y-2 justify-center flex-col absolute top-2 right-2 cu
                            <Link href={`product/${product.slug}`} className='⬜bg-white p-2 rounded-full'>
                                <Image className='group' src={eye} alt="Details image" width={18} height={19} />
                            </Link>
                            <p className="absolute top-full mt-1 left-1/2 -translate-x-1/2 text-xs ▨text-gray-600
                                details
                            </p>
                            <div>
                                <WishListButton product={product} />
                            </div>
                        </div>
                    </div>

                    {/* Product Details */}
                    <div className="detail flex flex-col p-4">
                        <p className='truncate 🟫text-[#9E3500] font-semibold text-sm'>{product.status}</p>
                        <div className="productDetail flex items-center justify-between">
                            <p className="truncate font-semibold text-sm">
                                {product.productName || 'Unnamed Product'}
```

3. Schema Code (Sanity CMS Structure)

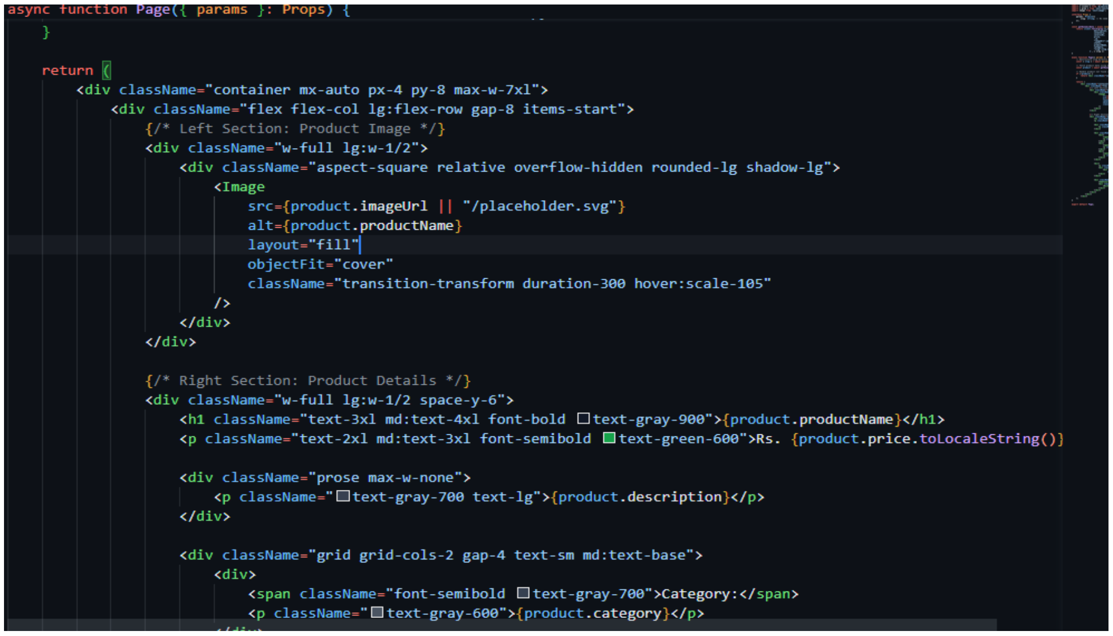
- Defines how **furniture product data** is stored in Sanity.
- **Key Fields:**
    - **Title** (Product name)
    - **Slug** (Unique URL identifier)
    - **Description** (Detailed product info)
    - **Price** (Numeric value with validation)
    - **Image** (Supports high-resolution uploads)
- **Validations:**
    - Ensures required fields are not empty.
    - Prices must be positive numbers.

- 

```
export const product = {
    name: 'product',
    title: 'Product',
    type: 'document',
    fields: [
      {
        name: 'productName',
        title: 'Product Name',
        type: 'string',
      },
      {
        name: 'category',
        title: 'Category',
        type: 'string',
      },
      {
        name: 'price',
        title: 'Price',
        type: 'number',
      },
      {
        name: 'inventory',
        title: 'Inventory',
        type: 'number',
      },
      {
        name: 'colors',
        title: 'Colors',
        type: 'array',
        of: [{ type: 'string' }],
      },
      {
        name: 'status'
```

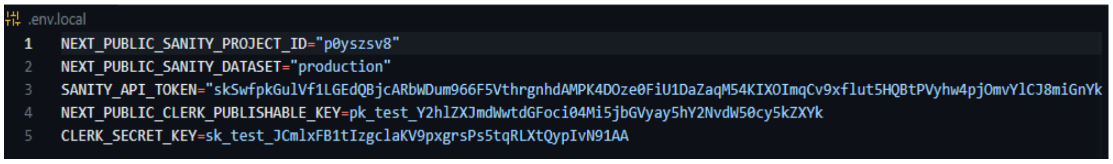
4. Item Card Code (Reusable UI Component)

- **Displays individual products** dynamically.
- Uses **props** (title, description, price, image) for flexibility.
- **Features:**
  - Add-to-cart button
  - Optimized images using `next/image`
  - Styled using **CSS or Tailwind CSS**
  - **Fully responsive**

- 

```
async function Page({ params }: Props) {
    }

    return (
        <div className="container mx-auto px-4 py-8 max-w-7xl">
            <div className="flex flex-col lg:flex-row gap-8 items-start">
                {/* Left Section: Product Image */}
                <div className="w-full lg:w-1/2">
                    <div className="aspect-square relative overflow-hidden rounded-lg shadow-lg">
                        <Image
                            src={product.imageUrl || "/placeholder.svg"}
                            alt={product.productName}
                            layout="fill"
                            objectFit="cover"
                            className="transition-transform duration-300 hover:scale-105"
                        />
                    </div>
                </div>

                {/* Right Section: Product Details */}
                <div className="w-full lg:w-1/2 space-y-6">
                    <h1 className="text-3xl md:text-4xl font-bold text-gray-900">{product.productName}</h1>
                    <p className="text-2xl md:text-3xl font-semibold text-green-600">Rs. {product.price.toLocaleString()}

                    <div className="prose max-w-none">
                        <p className="text-gray-700 text-lg">{product.description}</p>
                    </div>

                    <div className="grid grid-cols-2 gap-4 text-sm md:text-base">
                        <div>
                            <span className="font-semibold text-gray-700">Category:</span>
                            <p className="text-gray-600">{product.category}</p>
```

5. Environment Variables (.env Security)

- **Stores sensitive configurations** securely:
    - SANITY_PROJECT_ID (Project identifier)
    - SANITY_API_TOKEN (Used for secure API calls)
- **Security Best Practices:**
    - Not exposed to the frontend.
    - Accessed using `process.env`.
    - 

```
.env.local
NEXT_PUBLIC_SANITY_PROJECT_ID="p0yszsv8"
NEXT_PUBLIC_SANITY_DATASET="production"
SANITY_API_TOKEN="skSwfpkGulVf1LGEdQBjcARbWDum966F5VthrgnhdAMPK4DOze0FiU1DaZaqM54KIXOImqCv9xflut5HQBtPVyhw4pjOmvY1CJ8miGnYk
NEXT_PUBLIC_CLERK_PUBLISHABLE_KEY=pk_test_Y2hlZXJmdWwtdGFoci04Mi5jbGVyay5hY2NvdW50cy5kZXYk
CLERK_SECRET_KEY=sk_test_JCmlxFB1tIzgclaKV9pxgrsPs5tqRLXtQypIvN91AA
```

6. Explanation & Scalability Considerations

- Schema allows **future expansions** (e.g., adding stock levels, dimensions, materials).
- Data is **easily accessible via GROQ queries**, ensuring smooth frontend integration.
- **Validation rules** help maintain clean and consistent data.

7. Conclusion (Key Achievements on Day 3)

✅ **Data migration** from **Sanity CMS** to **Nike store database** completed.

✅ **Client-side UI** successfully fetched and displayed data dynamically.

✅ **Schema structured** for better organization and future scalability.

✅ **Reusable Item Cards** developed for rendering product listings.

✅ **Secure API integration** using environment variables.