Dynamic Frontend Components

1. Overview

This document provides a comprehensive guide on the implementation of the **Product Detail Page** and **Product Listing Page** in Next.js, integrated with Sanity CMS. It covers dynamic product fetching, related products, search functionality, pagination, and performance optimizations.

2. Product Detail Page

Key Features:

✓ Fetches individual product details dynamically from Sanity CMS using `slug`.

✓ Displays product image, name, price, discount, description, and stock level.

✓ Provides **Add to Cart** and **Buy Now** functionalities with SweetAlert notifications.

✓ Shows **Related Products** from the same category for enhanced user engagement.

Implementation Steps:

1. **Fetching Product Data:

- Uses `useParams()` to extract the `slug` from the URL.

- Calls `getProduct(slug)` to fetch product details from Sanity CMS.

- Calls `getRelatedProducts(category, currentProductId)` to fetch related products.

2. Displaying Product Details:

- Uses **Next.js Image component** for optimized image rendering.

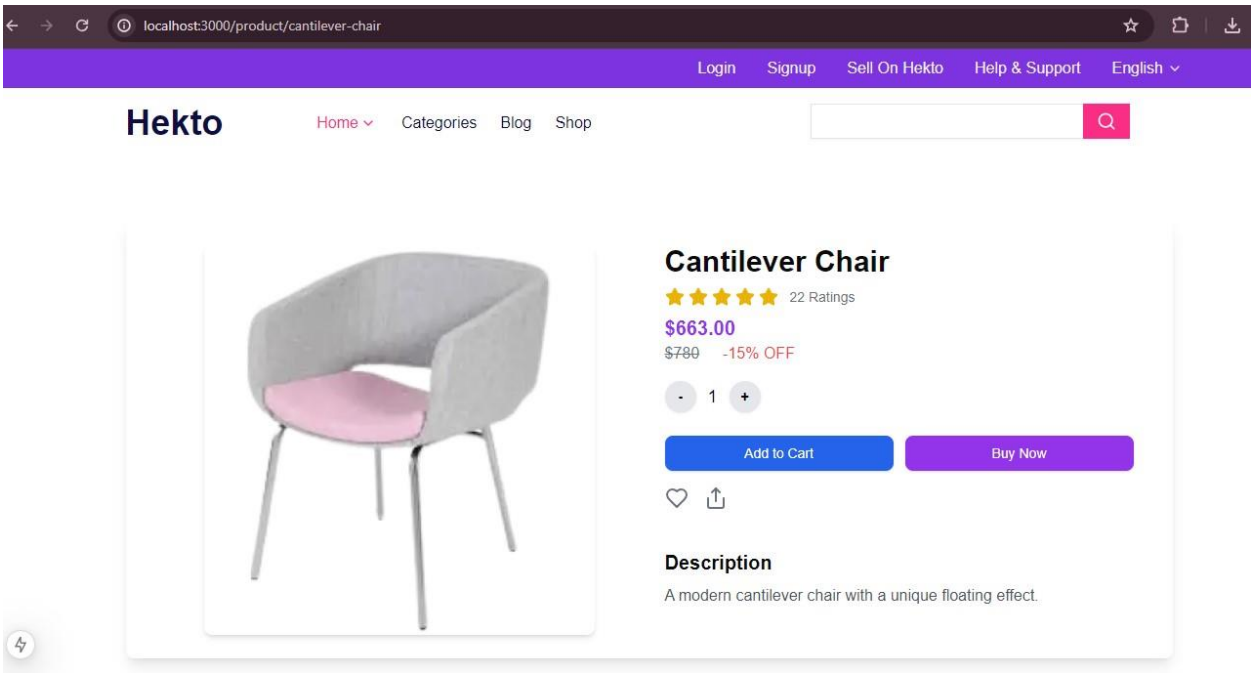
- Displays product name, price, discount, and stock information.

- Implements **quantity selection** with `+` and `-` buttons.

3. Related Products Section:

- Fetches and displays up to **4 related products** based on the same category.
- Provides clickable links to navigate to individual related product pages.

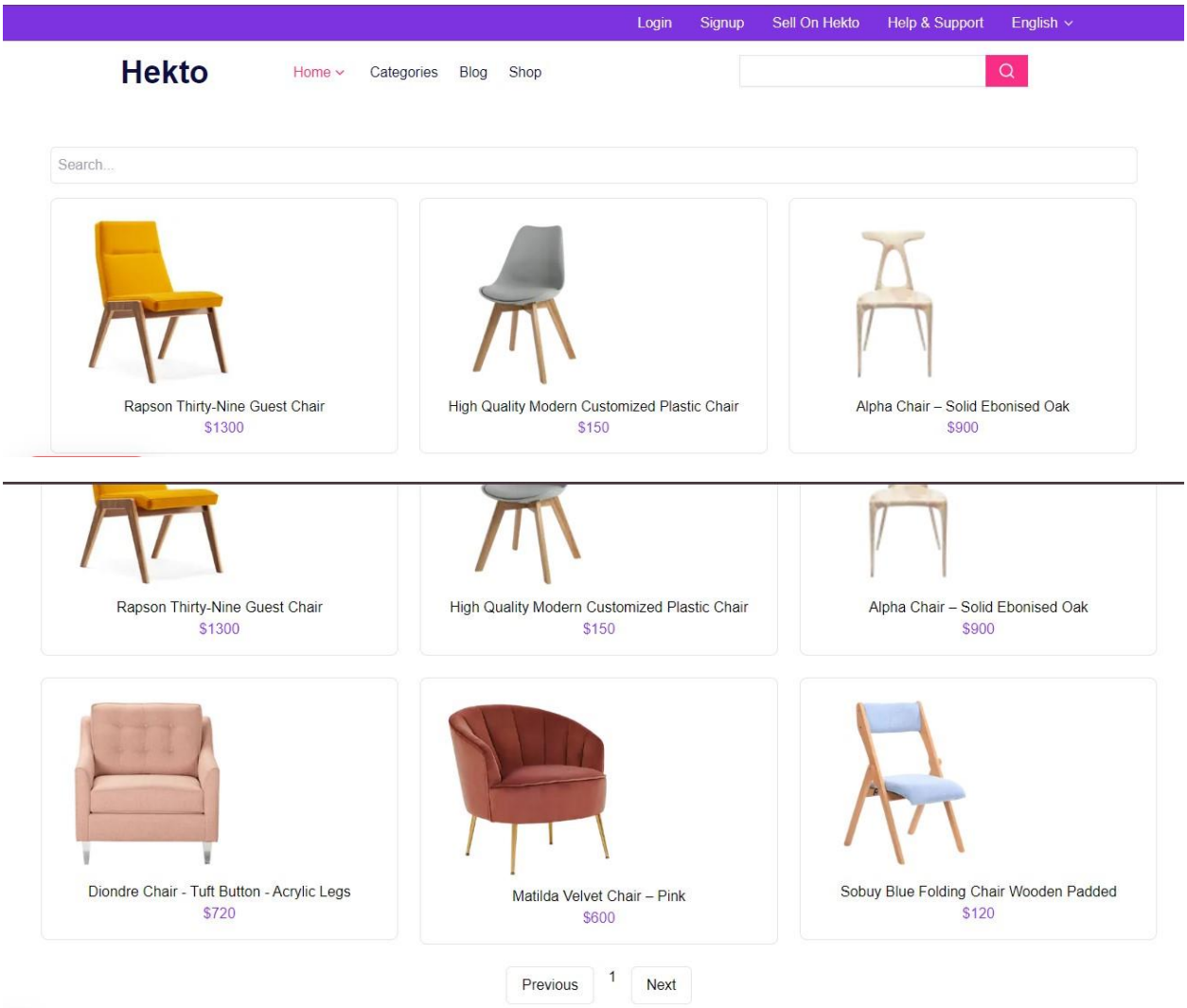
Code Snippet (Fetching Product Data)




```tsx
async function getProduct(slug: string): Promise<Product> {
  return client.fetch(
    groq`*[_type == "product" && slug.current == $slug][0]{
      _id, name, image, price, description, discountPercentage, stockLevel, category
    }`,
    { slug }
  );
```

}
```

---

3. Product Listing Page

Key Features:

✓ Fetches and displays all products from Sanity CMS dynamically.

✓ Implements **search functionality** to filter products by name.

✓ Uses **pagination** to optimize performance and prevent long product lists.

✓ Provides a responsive, user-friendly grid layout.

Implementation Steps:

1. Fetching Products:
   - Calls `client.fetch(allProducts)` to retrieve all products from Sanity CMS.
   - Stores the products in state using `useState()`.

2. Filtering Products:
   - Implements a **search bar** that allows filtering by name.
   - Uses `useState()` to track the search term and filters products dynamically.

3. Pagination:
   - Displays only a **limited number of products per page** (6 per page).
   - Implements **Previous** and **Next** buttons to navigate between pages.
```

Code Snippet (Filtering & Pagination)




```tsx
const filteredProducts = products.filter((p) =>
  p.name.toLowerCase().includes(search.toLowerCase())
);

const paginatedProducts = filteredProducts.slice(
  (currentPage - 1) * productsPerPage,
  currentPage * productsPerPage
```

);
```

---

4. Technologies Used

- **Next.js** – Handles routing and server-side rendering.

- **Sanity CMS** – Stores and manages product data.

- **Tailwind CSS** – Ensures a fully responsive design.

- **SweetAlert2** – Displays user-friendly notifications.

- **GROQ Queries** – Fetches optimized product data from Sanity CMS.

---

5. Future Enhancements

📌 **Category-Based Filtering** – Allow users to filter products by category.

📌 **Price Sorting** – Implement sorting (Low to High, High to Low) for better user experience.

📌 **Infinite Scrolling** – Improve pagination by adding a **Load More** button for smoother navigation.

---

This document serves as a complete guide to understanding the implementation of product pages in **Next.js with Sanity CMS integration**. 🚀
```