

Bilqees Shahid

00426703

Friday: 9am to 12pm

Mentor: Sir Hamzah Syed

DAY 6 - BUILDING DYNAMIC FRONTEND COMPONENTS FOR MARKETPLACE

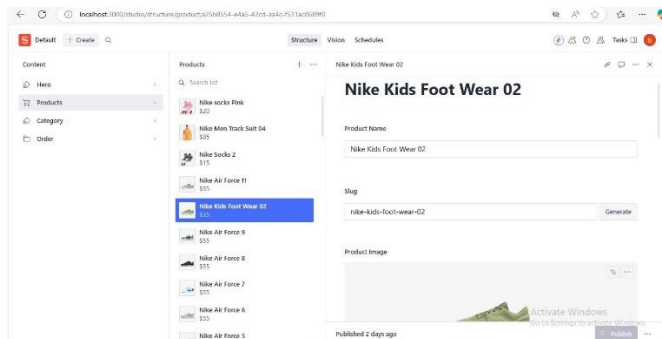
Hackathon Task Description: Day 6

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

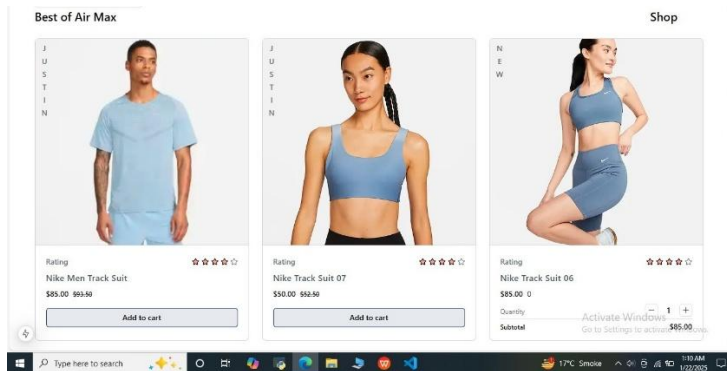


Sanity CMS was used to manage and organize the marketplace data efficiently. The process involved migrating product details, categories, user information, and other marketplace data into Sanity's schema. This step required creating a structured schema in Sanity Studio to define data types such as Products, Categories, 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:

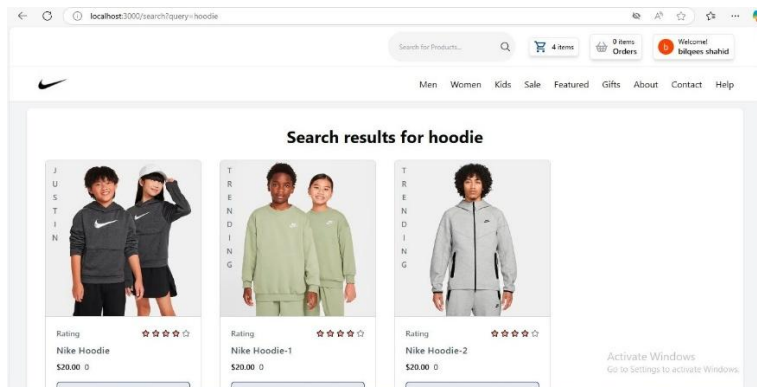
- Flexible content modeling.
- Real-time collaboration and updates.
- Scalable and highly structured data management.

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 in sanity:

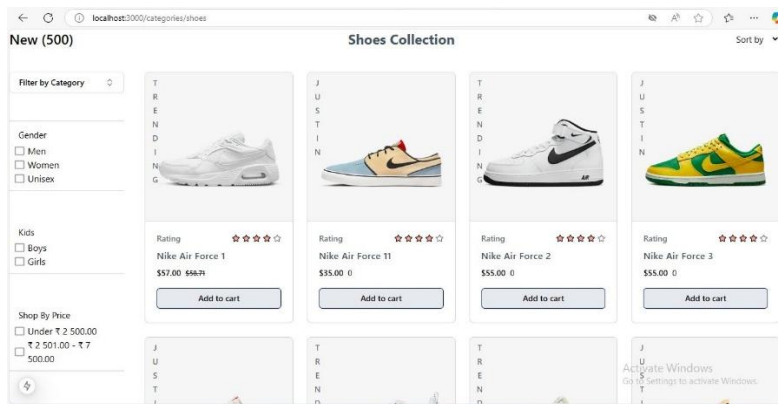
Product Listing with Sanity Integration

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.

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.

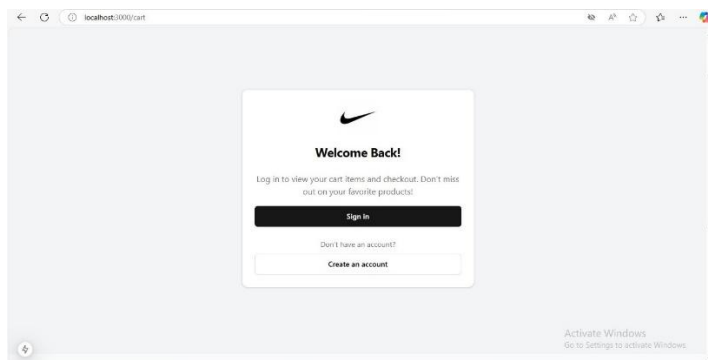


Sanity Schema for Product Details:

- 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

Authentication:

I have used the clerk's authentication for user to move towards cart page to see the order summary of the products purchased.



Dynamic Routing in Next.js

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

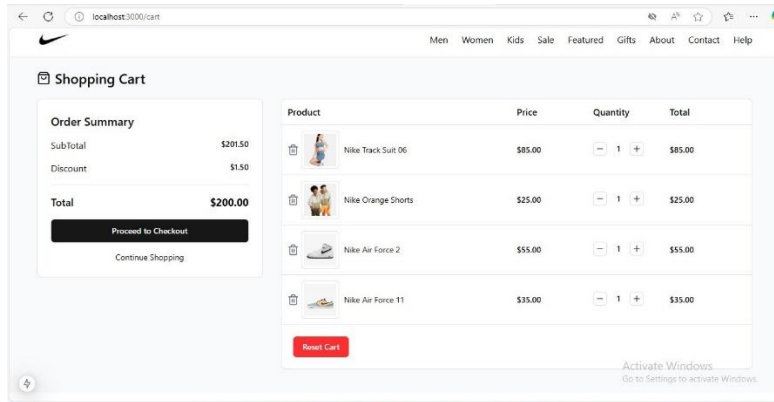
Rendering Product Details

- 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.

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.

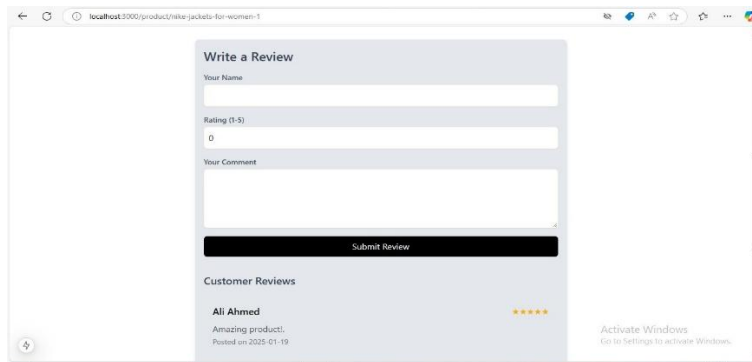
1. **Order Summary:** ○ Displays a detailed list of items in the cart, including product names, quantities, prices, and the total amount.
2. **User Information Form:** ○ Collects essential details such as name, address, phone number, and email for delivery purposes.
3. **Payment Options:** ○ Provides 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.

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 of 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.

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:

o 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.

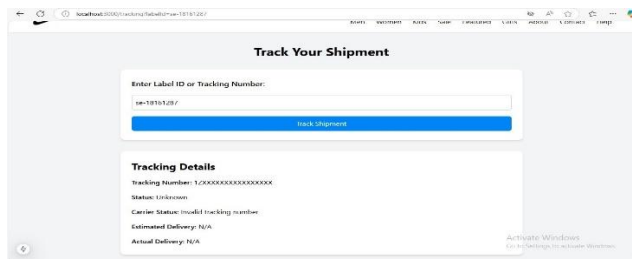
o The user can then choose their preferred shipping method before proceeding to payment.

3. Shipment Tracking: o After the order is placed and the shipment is processed, customers can track their packages using tracking numbers generated by ShipEngine.

o A tracking link is provided to the customer, allowing them to monitor the progress of their shipment in real-time.

4. Shipment Status Updates: o Customers receive notifications regarding their shipment status, including when it's dispatched, in transit, and delivered.

- o Any delays or issues are also communicated to customers to keep them informed throughout the delivery process.



Tracking:

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