Marketplace Technical Foundation -

Furniture Marketplace

I start my e-commerce website on the latest version of Next.js in the following way.

The basic pages of a website typically include the following

Homepage: The main landing page that introduces the website and its purpose.

About Us: A page that explains the background, mission, and values of the website or company.

Products/Services: A page listing the items or services offered for sale or use.

Contact Us: A page with contact information and a form to reach the website's team.

Signup/Login: Pages where users can register or log in to access personal features and make purchases.

Shopping Cart: A page where users can view and modify the items they plan to buy.

Checkout: The page where users finalize their purchase by entering payment and shipping details.

Terms & Conditions/Privacy Policy: Pages outlining the rules, regulations, and privacy practices of the website.

Each of these pages typically includes a Header (which usually contains the website logo, navigation links, and sometimes a search bar) and a Footer (which contains important links such as privacy policy, terms of service, and contact information).

First Step

Sign-up Page : The user will be shown the signup page upon opening the website. Once they

log in, their data can be stored in Sanity.

Description: Adding a login page to a website has multiple benefits

1. User Authentication: By implementing a login page, you can ensure that only authenticated users can access certain parts of the website, providing a more secure experience.

2. Security: A login page adds a layer of security, preventing unauthorized access to sensitive areas or information.

When the website is opened, the user should be presented with a signup page where they are given permission to enter the website and proceed with shopping.

We use signup to ensure that only authorized users can access the website and make purchases. It helps in personalizing the experience, tracking orders, and managing customer data securely.

Step-2

1. Data Storage:

Sanity APIs allow you to store structured data, such as blog posts, product details, or user profiles. This data is organized in Sanity Studio, which works like a CMS and makes content management easy.

2. Easy Data Fetching:

Fetching data from Sanity is simple using GraphQL or REST APIs. These APIs are fast and flexible, enabling you to retrieve dynamic content for front-end applications like React or Next.js efficiently.

3. Real-Time Updates:

Sanity provides real-time updates, meaning changes in content are instantly reflected on the front-end without requiring a page refresh. This feature is perfect for collaborative and dynamic applications.

Step-3

Product Browsing:

User views product categories -> Sanity API fetches data -> Products

displayed on frontend.

step-4

Order Placement:

User adds items to the cart -> Proceeds to checkout -> Order details

saved in Sanity.

Step-5

Proceed to Checkout Functionality with Sanity

The "Proceed to Checkout" feature is commonly used in e-commerce applications. Its primary purpose is to send user order data to Sanity CMS for management and processing.

Frontend Data Collection: Gather order details such as product information, quantity, price, and user data when the user clicks the checkout button.

Sanity API Integration: Use the @sanity/client package to send the collected data to Sanity CMS.

Sanity Schema for Orders: Define a schema in Sanity Studio to store order data, including fields like user name, email, items, and total amount.

Save Order to Sanity: Use the create method from Sanity client to save the order data into the specified schema.

Real-Time Feedback: Provide feedback to the user, such as a confirmation message or error alert, based on the API response.

Step-6

APIs provided by third parties can offer various functionalities that help enhance your website or application. Here's an example of such APIs:

Payment Gateway API:

Example: Stripe API or PayPal API

Use: These APIs allow you to securely process payments on your website, enabling users to pay for products or services.

Step-7

Shipment Tracking:

o Order status updates fetched via 3rd-party API -> Displayed to the user.

Responsive Design

Meaning and Importance of Responsive Design

Responsive design ensures that a website fits and functions well on all screen sizes (mobile, tablet, desktop). This technique uses tailwindCSS, CSS media queries and flexible grids to adjust the layout according to the screen size.

Benefits :

Better User Experience: Provides seamless viewing and usability across all devices.

SEO Friendly: Google ranks responsive websites higher in search results.

Time and Cost Saving: One website works on all screen sizes, eliminating the need for separate designs.

APIs:

APIs: How They Work and Endpoints Overview

APIs (Application Programming Interfaces) allow communication between a client (e.g., front-end) and a server (e.g., database or backend). APIs define endpoints—URLs that perform specific actions using HTTP methods. Below is a breakdown of API creation, endpoint setup, and methods with their descriptions.

Steps to Create APIs

1.Backend Setup:

Use frameworks like Node.js (with Express.js), Django, or Flask to build your API.

2.Define Endpoints:

Set URLs for different operations like fetching data, posting new data, updating, or deleting records.

3.Connect Database:

APIs interact with databases (e.g., MongoDB, Sanity, PostgreSQL) to retrieve or modify data.

4.Authentication:

Secure APIs using tokens (e.g., JWT) or API keys to ensure only authorized users can access data.

HTTP Method	Description	Example
GET	Fetch data from the server.	/products to get all products.
POST	Submit new data to the server.	/orders to create a new order.
PUT	Update existing data entirely.	/products/123 to update product.
PATCH	Partially update existing data.	/users/45 to update user email.
DELETE	Remove data from the server.	/orders/567 to delete an order.

---

1. Products API

GET /products: Retrieve a list of all products.

GET /products/:id: Retrieve details of a specific product by ID.

POST /products: Add a new product.

PUT /products/:id: Update a product completely by ID.

DELETE /products/:id: Delete a product by ID.

2. Orders API

GET /orders: Fetch all orders.

POST /orders: Create a new order with customer and product details.

GET /orders/:id: Retrieve a specific order's details.

DELETE /orders/:id: Cancel an order.