Technical Architecture for Furniture E-Commerce Store...

This document provides a comprehensive overview of the technical architecture for your furniture e-commerce platform, designed to showcase and sell high-quality furniture products, including wood, sofas, chairs, tables, and more. The platform leverages **Next.js** for a dynamic frontend experience, **Tailwind CSS** for modern UI styling, **Sanity CMS** for efficient product and content management, and **ShipEngine** for streamlined shipment tracking and delivery management.

Frontend & Styling:

Framework: Built using **Next.js**, a React-based framework for server-side rendering and static site generation, ensuring fast load times and SEO optimization.
Styling: Styled using **Tailwind CSS**, a utility-first CSS framework for modern UI design with minimal code.

Data Handling:

- The frontend interacts with the backend through **RESTful APIs**.
- Data such as products, orders, and user information is fetched, updated, and deleted using API requests.

Backend:

The backend manages the core functionality for product, order, and user data handling.

Sanity CMS Integration:

- **Sanity CMS** serves as the content management system and database for:
  - **Product Data:** (title, description, price, stock, category, images)

    - **Order Data:** (order ID, user details, products, order status)
    - **User Data:** (user ID, role, credentials)
- **Real-Time Sync:** Any content change is reflected instantly in the frontend due to the headless CMS architecture.

The backend manages the core functionality for product, order, and user data handling.

Sanity CMS Integration:

- **Sanity CMS** serves as the content management system and database for:
    - **Product Data:** (title, description, price, stock, category, images)
    - **Order Data:** (order ID, user details, products, order status)
    - **User Data:** (user ID, role, credentials)
- **Real-Time Sync:** Any content change is reflected instantly in the frontend due to the headless CMS architecture.

Database:

- **Type:** NoSQL database using **Sanity CMS**.

Purpose: Flexible data storage for:

- Users (IDs, roles, credentials)
- Products (metadata, images, stock)
- Orders (order history, status, payment info)

Product Management (API Integration):

APIs allow efficient CRUD (Create, Read, Update, Delete) operations for products stored in the **Sanity CMS**.

Fetching Products (GET Request)

- **Endpoint:** `GET /api/products`
- **Purpose:** Retrieve all products from the CMS to display on the store.

Add New Product (POST Request)

- **Endpoint:** `POST /api/products`
- **Authorization:** Requires `Admin` or `Seller` role.
- **Purpose:** Add a new product with properties like `title`, `price`, `stock`, and `category`.

Update Product (PUT Request)

- **Endpoint:** `PUT /api/products/:id`
- **Authorization:** Requires `Admin` or `Seller` role.
- **Purpose:** Update an existing product's details by its unique ID.

Delete Product (DELETE Request)

- **Endpoint:** `DELETE /api/products/:id`
- **Authorization:** Requires `Admin` or `Seller` role.
- **Purpose:** Remove a product from the CMS using its unique ID.

Order Management (API Integration):

The order management system ensures secure and trackable transactions between customers and the store.

Create New Order (POST Request)

- **Endpoint:** `POST /api/orders`
- **Purpose:**
  - Generate a new order in **Sanity CMS** with user and product details.
  - Required fields: `user_id`, `product_ids`, `quantity`.

Payment Management (API Integration):

A secure payment flow integrated through API endpoints.

- **Initiate Payment:**
  - **Endpoint:** `POST /api/payments`
  - **Purpose:** Start a new payment process for a placed order.
- **Check Payment Status:**
  - **Endpoint:** `GET /api/payments/status`
  - **Purpose:** Fetch the status of an ongoing payment.

Fetch All Orders (GET Request)

- **Endpoint:** `GET /api/orders`
- **Purpose:** Fetch a list of all orders placed by the authenticated user.

Fetch Specific Order (GET Request)

- **Endpoint:** `GET /api/orders/:id`
- **Purpose:** Retrieve details of a specific order, including product list, order status, and payment history.

Order Status Management:

- **Order Status Fields:** `pending`, `confirmed`, `shipped`, `delivered`.
- **Status Tracking:**
  - **Endpoint:** `GET /api/orders/:id/status`
  - **Purpose:** Fetch the current order status for real-time tracking.

Shipment Management (ShipEngine Integration):

ShipEngine is used to manage order shipments, track delivery, and provide real-time shipment updates.

Workflow Using ShipEngine API:

1. **Order Creation:**
    - Once an order is placed, a shipment request is automatically created using the **ShipEngine API**.
    - **Endpoint:** `POST /api/shipments`
    - **Action:** Creates a shipment with customer and order details sent to ShipEngine.
2. **Generate Shipping Label:**
    - **Endpoint:** `POST /api/shipments/label`
    - **Action:** A shipping label is generated for the order using ShipEngine and stored in **Sanity CMS**.
3. **Track Shipment:**
    - **Endpoint:** `GET /api/shipments/track/:trackingNumber`
    - **Action:** Fetch real-time tracking details from **ShipEngine**.
4. **Update Shipment Status:**
    - ShipEngine automatically updates the shipment status (`pending`, `in transit`, `delivered`) in **Sanity CMS**.

Component Details and Interaction:

A breakdown of how the frontend and backend components interact.

Frontend (Next.js):

- **Role:** Handles UI rendering and user interactions.
- **Responsibilities:**
    - Fetch data from the backend via APIs.
    - Manage user actions like adding products, placing orders, and tracking shipments.

Backend (Sanity CMS + API Layer):

- **Role:** Acts as the content database and API handler.
- **Responsibilities:**
    - Provide RESTful APIs for products, orders, users, and shipments.
    - Ensure secure data handling and real-time content updates.