# Furniture Website System Architecture

Name: Bilal Muhammad Ashraf

**Overview**

This document outlines the system architecture of an e-commerce platform focused on affordable and trendy Pinterest-inspired furniture. The platform is developed using Next.js 14 with TypeScript and leverages Sanity CMS for content management. It incorporates various pages, workflows, and technologies to ensure a smooth user experience, robust admin functionalities, and efficient data management.

High-Level System Architecture

Frontend Structure

**Framework**:

Next.js 14 with TypeScript for server-side rendering (SSR) and fast dynamic routes.

**Pages**:

General Pages: Home, About, Products, Product Details (dynamic), Cart, Admin Panel (admin-only access).

User Pages: Login, Sign Up, User Portal (order and shipment details).

Admin Pages: Analytics, Dashboard, Orders, Stock Management, Users.

**API Endpoint**:

Product API:https://hackathon-apis.vercel.app/api/products

Reusable Components

**UI Components**:

CardComponent.tsx, Feature.tsx, HeroSection.tsx, Listing.tsx, PopularProduct.tsx – Ensure consistent design and reusable patterns across the platform

**Product Components:**

ProductComponent.tsx – Displays product details.

ProductCardDetails.tsx – Provides a detailed view of individual products.

**Order Components:**

CheckoutModal.tsx – Collects user information during checkout.

PaymentForm.tsx – Integrates Stripe for payment processing.

DisplayShipmentDetails.tsx – Provides shipment tracking details.

**Cart Components:**

UserCartComponent.tsx – Manages cart data stored locally.

CartItem.tsx – Displays individual cart items.

**User Authentication:**

UserLogin.tsx and UserSignup.tsx – Handle user login and signup processes.

Content Management System (Sanity CMS)

Sanity Studio manages dynamic content and structured data such as:

Products (name, price, images, categories, inventory levels).

Users (authentication and order history).

Orders (items, quantities, shipping information).

Shipments (tracks shipment statuses via Shippo API).

Analytics (sales performance, revenue, product popularity).

**Data Schemas:**

Define structures for Products, Orders, Users, Inventory, and Analytics.

**GROQ Queries:**

Enable real-time data fetching from Sanity for display on the frontend.

Payment Gateway (Stripe)

**Purpose:**

Securely manage payments and simulate real-world transactions.

**Key Features:**

Stripe Elements for securely collecting payment details.

Dummy transaction processing for testing purposes.

Shipment Tracking (Shippo API)

**Purpose:**

Provide real-time shipment tracking information to users.

**Key Features:**

Live tracking on the user's order history page.

Fetch accurate shipment data post-payment confirmation.

**Workflow Overview**

### User Workflow:

Visit Home Page:

Browse product categories dynamically fetched from Sanity via mock APIs.

**Add to Cart:**

Items stored locally without requiring login.

**Checkout Process:**

Prompts login/signup before collecting shipping and payment information.

**Payment Processing:**

Secure transaction via Stripe.

**Shipment Tracking:**

Shipment request generated via Shippo API and tracking details displayed.

### Admin Workflow:

**Login:**

Access admin functionalities via AdminLogin.tsx.

**Analytics:**

View sales and order performance via Analytics.tsx.

**Inventory & Order Management:**

Manage stock via Stock.tsx and oversee orders via Orders.tsx.

**Navigation:**

Use SideBar.tsx for seamless admin navigation.

Challenges Faced

**Schema Design:**

Developing efficient schemas in Sanity for Products, Users, and Orders was challenging but refinements improved efficiency.

**API Integration:**

Extensive debugging was required to integrate mock APIs with the frontend.

**Workflow Implementation:**

Mapping workflows for cart, payment, and shipment tracking required significant planning.

Feedback and Adaptations

**User Testing:**

Improvements made to UserCartComponent.tsx for better usability.

**Admin Feedback:**

Enhanced analytics visuals with changes to Analytics.tsx.

Learning Outcomes

Improved understanding of e-commerce system architecture.

Enhanced TypeScript, Next.js, and CMS integration skills.

Practical experience in API design and seamless user experiences.

Technologies and Tools

**Frontend:** Next.js 14 with TypeScript.

**CMS**: Sanity (content and data management).

**Payment Gateway:** Stripe (payment processing).

**Shipment Tracking**: Shippo API (live shipment updates).

**Deployment:** Hosted on Vercel for fast and reliable

delivery.

Prepared by: Bilal