System Architecture$_{D}ay2$

Aneeq Ahmed

January 2025

Marketplace Technical Foundation - [Hekto] 1. System Architecture Overview Introduction Our e-Commerce marketplace architecture will be designed such that the interaction between the user interface, the content management system, and the third-party services is smooth and seamless. For the frontend, the system shall be powered by Next.js; the backend, Sanity CMS, and third-party APIs for payment, shipment tracking, and product data. The system architecture diagram below is the high-level system architecture showing where and how data flows between components.

scss/ Copy Frontend (Next.js) — [Sanity CMS] — [3rd Party APIs] / — Payment Shipment Product API Component Overview Frontend (Next.js): The frontend will display a user interface for viewing and browsing products and adding items into the cart or completing orders, which will further interact with the backend via APIs. Sanity CMS: The sanity will keep all product details, customer's information, and also order records; it will also act as the database for this marketplace and be accessed by frontend via APIs 3rd Party APIs: Payment Gateway API used to process a secure payment in case of successful orders. Shipment Tracking API: Returns real-time shipment tracking information. Product Data API: Retrieves product data from Sanity CMS and the front-end renders it. Data Flow: Product Browsing: The front-end interacts with the user, which fetches product data from Sanity CMS through an API. Order Placement: Once an order is placed, the details are sent to Sanity CMS through an API call. Payment Processing: The payment gateway processes secure transactions. Shipment Tracking: Once a new order is requested, shipment information is retrieved through the Shipment Tracking API 2. Main Flows 1. User Surfs for Products On the marketplace's home page, he opens and sees product types The frontend makes a GET request to the /products endpoint which retrieved all the information needed for the products in the Sanity CMS On the user's screen, the products are dynamically shown including names, prices, and images 2. User Adds Items to Cart The user chooses his products and moves them to a shopping cart. The cart is temporarily stored at the frontend in order to manage the items to be ordered by using sessionStorage or Redux. There is a summary for the cart provided to the user, which could be edited and modified. 3. User Creates an Order When the user is ready to check out, he reviews his cart and proceeds to click "Proceed to Checkout". The user submits their shipping information and payment details. The front-end makes

a POST request to the /orders API endpoint, which creates the order in Sanity CMS and returns an order ID. The order status is updated and a confirmation message is sent to the user. 4. Shipment Tracking Following the creation of the order, the Shipment Tracking API is called to fetch the tracking details of the order. The front-end shows the status of the order; for example, "Shipped", "In Transit", or "Delivered", along with an estimated delivery date. 3. API endpoints 1. /products (GET) Fetch all the available product details from Sanity CMS. Response Example: json Copy ["id": 1, "name": "Product A",

"price": 100, "image": "url to image", "stock": 50, "category": "Electronics" ,

"id": 2, "name": "Product B", "price": 200, "image": "$\text{url}_t o_i mage$","$stock$" : 30,"$category$" : "$Fashion$"]2.$/orders(POST)$

Purpose: Creates a new order in Sanity CMS with customer and product information.

example:

json

Copy " customerId": 321,

" products": [" productId": 1, "quantity": 2 ,

"productId": 2, "quantity": 1],

" totalAmount": 400, "status": "Pending" Response Example:

json

Copy

" orderId": 101, " status": "Success", "confirmationId": 12345 3. /shipment (GET) Purpose : fetches shipment tracking information from a third-party API. Response Example: json Copy "shipmentId": 123, "orderId": 101, "status": "In Transit", "ETA": "2 days" 4. Sanity Schema Example Product Schema javascript Copy export default name: 'product', type: 'document', fields: [name: 'name', type: 'string', title: 'Product Name' , name: 'price', type: 'number', title: 'Price' , name: 'stock', type: 'number', title: 'Stock Level' , name: 'image', type: 'image', title: 'Product Image' , name: 'category', type: 'string', title: 'Category' name: 'description', type: 'text', title: 'Product Description']; ; Order Schema javascript Copy export default name: 'order', type: 'document', fields: [name: 'customerName', type: 'string', title: 'Customer Name' , name: 'totalAmount', type: 'number', title: 'Total Amount' , name: 'status', type: 'string', title: 'Order Status' , name: 'products', type: 'array', of: [type: 'reference', to: [type: 'product']] , name: 'paymentStatus', type: 'string', title: 'Payment Status' , name: 'shippingAddress', type: 'string', title: 'Shipping Address'] ; 5. Technical Roadmap To complete the project within the stipulated time, here is the technical roadmap outlined below:

Phase 1: System setup Create the Next.js project for the front-end. Configuration in Sanity CMS - Schemas for product and orders. Phase 2: API development Create, test, API endpoints: (/products, /orders, /shipment) Implementation of third-party payment gateway: Integrating with some third-party service for the transaction. Phase 3: Development of the front-end Product browsing page: Design of a responsive web page to allow browsing products. Shopping Cart and Checkout Pages: Implement cart functionality and checkout

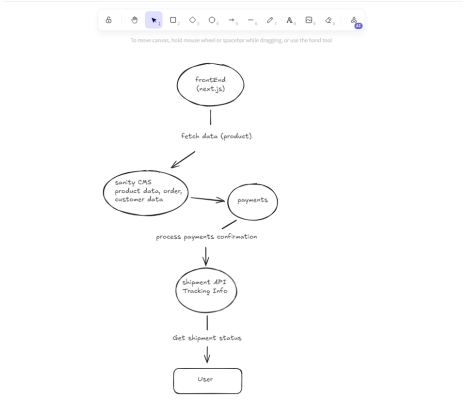

Figure 1: Enter Caption

flow. Order Confirmation and Shipment Tracking: Develop the order confirmation page and shipment tracking system. Phase 4: Testing Deployment Unit Testing and QA: Test the front- and back-end components for functionality and user experience.