“DAY 2”

BUILDING THE TECHNICAL FOUNDATION FOR MY GENERAL E-COMMERCE PLATFORM:

To build a successful e-commerce platform, it’s important to translate business goals into technical features. This ensures the platform is both functional and aligned with customer needs.

SOME TECHNICAL REQUIREMENT:

FRONTEND REQUIREMENTS:

The frontend will focus on providing a user-friendly interface that is simple and easy to navigate. A responsive design will ensure the platform works seamlessly on both mobile and desktop devices. Essential pages, such as Home, Product Listing, Product Details, Cart, Checkout, and Order Confirmation, will guide customers smoothly through their shopping journey. The platform will include the following **essential pages** to guide customers smoothly through their shopping journey:

- **Home Page**: The main landing page showcasing featured products, promotions, and categories.
- **Product Listing Page**: A page displaying a catalog of products with filtering and sorting options.
- **Product Details Page**: A detailed view of each product, including images, descriptions, pricing, and reviews.
- **Cart Page**: A page where customers can view selected products, update quantities, and proceed to checkout.
- **Checkout Page**: A page for customers to provide shipping details, choose payment methods, and confirm orders.
- **Order Confirmation Page**: A final page that confirms the order has been placed successfully and provides order details.
- **Contact Us Page**: A page for customer inquiries, support, and feedback.
- **About Us Page**: A page highlighting the company’s vision, mission, and team.
- **User Account Page**: A dashboard where customers can view their profile, order history, and saved items.

Next.js will be used to build the frontend due to its advanced features like server-side rendering (SSR) and static site generation (SSG), which improve page loading speeds and overall performance. Next.js also supports easy routing, making it simple to create a scalable and efficient multi-page website. Its built-in SEO capabilities will ensure that the platform ranks well in search engines, attracting more visitors.

SANITY CMS AS BACKEND:

Sanity CMS will be used as the backend system to manage all critical data, such as product information, customer details, and order records. By designing schemas in Sanity, the database structure will be tailored to meet the specific business goals. This allows for efficient data handling and easy updates to the platform content.

THIRD-PARTY API's:

APIs will be integrated to handle important backend services like shipment tracking and payment gateways. These APIs will ensure that the platform has real-time data for key functionalities, such as tracking orders and processing secure payments.

SYSTEM ARCHITECTURE FOR BETTER UNDERSTANDING:

This is how my system architecture is designed to make everything work smoothly:

1. **Frontend (Next.js):** It handles all user interactions, like browsing products, adding items to the cart, and placing orders.
2. **Sanity CMS:** This is the backend system where all product data, customer details, and order information are stored and managed.
3. **Product Data API**: It fetches product information from Sanity CMS and sends it to the frontend so users can see it.
4. **Third-Party APIs**: These are used for shipment tracking, providing real-time updates to users about their orders.
5. **Payment Gateway**: It securely processes payments and sends confirmation back to Sanity CMS and the users.

EXAMPLE SYSTEM ARCHITECTURE:

- **Frontend (Next.js)**: The platform where users browse, shop, and interact.
- **Sanity CMS**: The backend system that stores all the important data about products, customers, and orders.
- **Third-Party APIs**: External services that connect to the system for shipment tracking and other needs.

KEY WORKFLOWS:

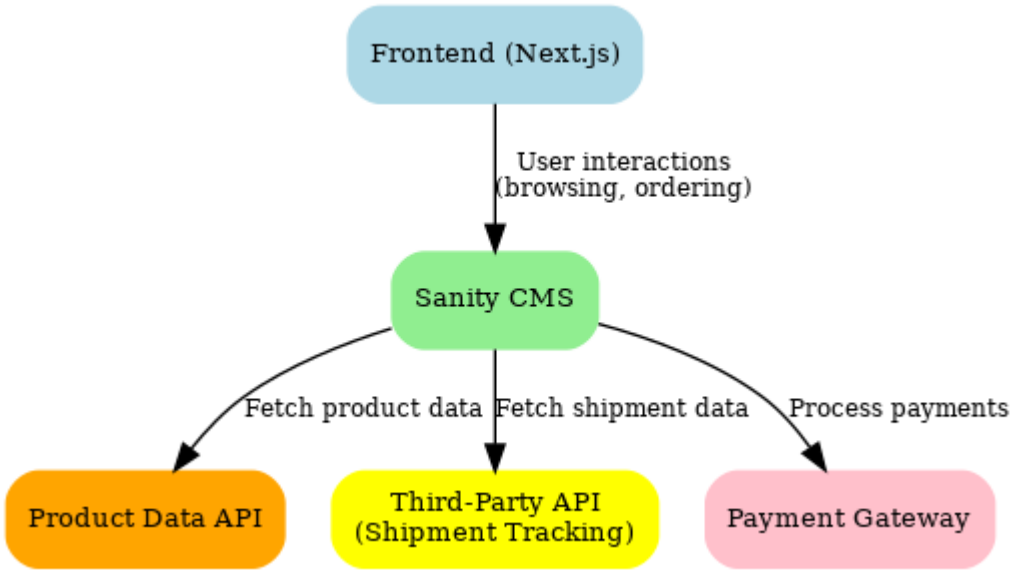
User Registration: A user signs up, their information is saved in Sanity CMS, and they receive a confirmation message.

Product Browsing: Users view product categories, and the data is fetched from Sanity CMS through an API, showing up-to-date details on the frontend.

Order Placement: A user adds products to the cart, proceeds to checkout, and their order details are stored in Sanity CMS.

Shipment Tracking: Updates about the order's delivery status are fetched using a third-party API and shown to the customer.

To better understand how these components interact, here's a diagram illustrating the data flow between them:



API DESIGN FOR GENERAL E-COMMERCE PLATFORM:

To make my e-commerce platform work smoothly, I've planned a set of APIs that handle important tasks like fetching products, creating orders, and tracking shipments. Here's how these APIs are set up:

1. **Endpoint Name: /products**
    - **Method**: GET
    - **Description**: This endpoint will fetch all available products from the Sanity CMS.
    - **Response**: It will return product details like ID, name, price, stock, and images.
2. **Endpoint Name: /orders**
    - **Method**: POST
    - **Description**: This is used to create a new order in the Sanity CMS.
    - **Payload**: It will include customer information, product details, and payment status.
3. **Endpoint Name: /shipment**
    - **Method**: GET
    - **Description**: This endpoint will track the order's shipment status using a third-party API.
    - **Response**: It will provide the shipment ID, order ID, status, and the expected delivery date.

HOW THESE APIS WORK TOGETHER

- The /products endpoint ensures that customers can view all available products dynamically on the platform.
- The /orders endpoint processes customer orders, saving all relevant details in Sanity CMS.
- The /shipment endpoint keeps customers updated with real-time tracking for their orders, enhancing their shopping experience.

This clear API structure ensures that the e-commerce platform functions efficiently while meeting customer needs.