Day # 02 *PLANNING THE TECHNICAL FOUNDATION*

Name: Alishba Shabbir

System Architecture

1. User Interface (Frontend - Next.js):

1. Authentication Module:

Handles user login, registration, and authentication processes.

2. Product Display Module:

Fetches and displays product data dynamically from the backend (Sanity CMS).

3. Order Management Module:

Manages the cart, checkout process, and order history display.

2. Backend System (Sanity CMS):

1. Product Management:

Stores product information such as product names, prices, categories, and stock levels.

2. Content Management:

Manages static content like promotional material, blog posts, or static pages.

3. Database Layer:

Managed by Sanity, this stores structured data like user information, product details, and order data.

4. **Asset Storage:**

Handles the storage of media assets, such as product images, banners, and downloadable files.

3. Third-Party APIs:

1. **Payment Gateway API:**

   Handles secure payment processing (e.g., Stripe, PayPal).

2. **Shipment Tracking API:**

   Tracks the shipment status and updates the user about the delivery progress.

4. API Requests and Responses:

Data exchange between the frontend and backend occurs through REST APIs or GraphQL.

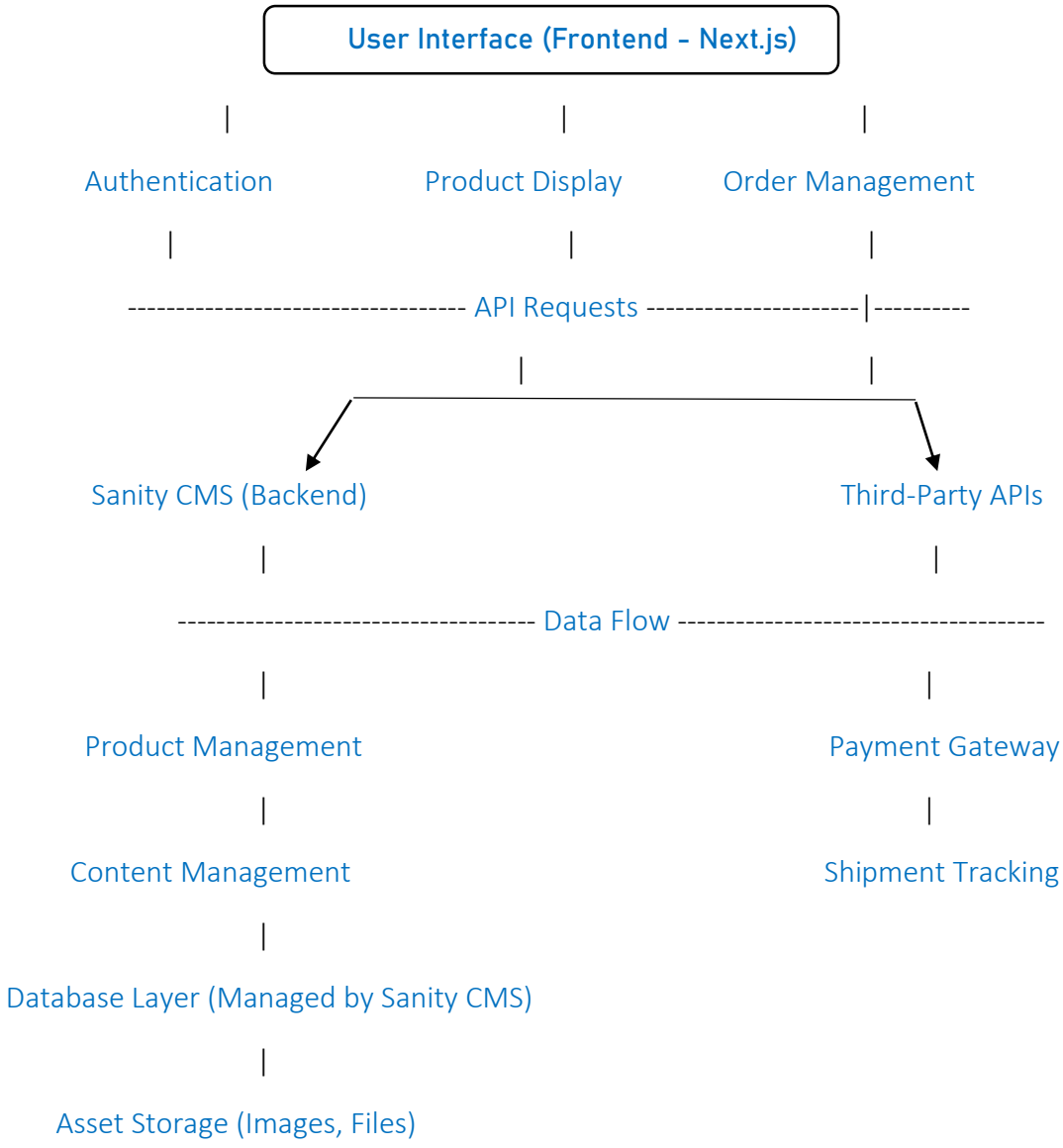
Key API Endpoints:

GET /products: Fetch product catalog.

POST /orders: Create an order.

GET /order-status: Fetch shipment and payment status.

System Architecture Diagram



5. Detailed Workflow Examples:

1. User Browsing Products:

The user selects a product on the frontend interface.

The frontend sends a GET request to the **Sanity CMS API** to retrieve product details.

Sanity CMS returns the product data, which is displayed on the user interface.

2. Placing an Order:

The user adds products to their cart and proceeds to checkout.

The frontend sends a POST request to **Sanity CMS** to store the order details.

Payment details are sent to the **Payment Gateway API**, which validates and processes the payment.

Upon successful payment, a shipment request is sent to the **Shipment Tracking API**.

3. Tracking an Order**:**

The user enters their order ID into the frontend interface.

The frontend sends a GET request to the **Shipment Tracking API**.

The shipment tracking status is returned and displayed to the user.
