Day 2 Hackathon

Marketplace Technical Foundation – Bandage

1. **Frontend (Next.js):**
    - **The Next.js application acts as the frontend where all user interactions occur, such as browsing products, filling out forms, and navigating pages.**
    - **It contains the UI components like ProductCard, ProductCard2, and ShipmentForm, which display data fetched from the backend or external APIs.**
2. **Sanity CMS:**
    - **Sanity CMS is used as the headless CMS to store and manage your product data, text content, images, and other assets.**
    - **Sanity Client is responsible for fetching this data and passing it to the Next.js application for rendering.**
    - **The fetched data includes product details such as headings, descriptions, prices, and images.**
3. **3rd Party APIs (Shipping API):**
    - **For shipment-related features, you might use third-party APIs (like USPS, UPS, or FedEx) to provide real-time shipping rates, tracking, or other logistical services.**
    - **APIs might also include order processing, payment gateways, and external services that add functionality to your platform.**
4. **UI Components:**
    - **These are React-based components that render content on the frontend:**
        - **ProductCard and ProductCard2 components display product data such as images, titles, prices, and descriptions, all fetched from the Sanity CMS.**

- **ShipmentForm handles user inputs for shipment details (e.g., address, shipping method) and sends the data to the 3rd Party APIs for processing.**

5. **Dynamic Routing:**
    - **Next.js Dynamic Pages handle routing dynamically based on product data (e.g., product detail pages with unique URLs) and shipment confirmation pages.**
    - **For instance, /products/[id] could be a dynamic route that displays a product's details.**
    - **Similarly, /shipment/[orderId] can be used for showing shipment confirmation or tracking.**
6. **Static and Dynamic Content:**
    - **Static content includes predefined pages or components, such as the product listing page.**
    - **Dynamic content refers to content that changes based on user interaction or data, like personalized product recommendations or shipment updates.**
7. **Client-Side Storage:**
    - **Client-side storage (like LocalStorage) is used to temporarily store data on the user's browser. For example, when a user fills out a shipment form, the data can be stored locally before being sent to the shipping API or backend.**
8. **User Authentication (Optional):**
    - **This component is optional and would involve implementing a login and authentication system.**
    - **You can use JWT (JSON Web Tokens) or session management for user authentication to secure certain parts of the site, like order history or profile pages.**

9. **Deployment:**
    - **The Next.js application is deployed on a platform like Vercel, Netlify, or AWS. These platforms provide automatic deployment from Git repositories and are optimized for Next.js applications.**
10. **Database (Optional):**
    - **Although you're using Sanity CMS for managing content, you might also need a database for managing user accounts, orders, or transaction data.**
    - **Sanity Dataset could store all your content, such as products, blog posts, and user-generated content.**
    - **For transactional data, you might use external databases like PostgreSQL, MongoDB, or Firebase.**

WORKFLOW

1. **Product Browsing:**
    - **User views product categories and details -> Data fetched from Sanity CMS -> Products displayed.**
2. **Product Details View:**
    - **User clicks on a product -> Data fetched from Sanity CMS -> Product details displayed.**
3. **Cart Addition:**
    - **User adds items to the cart -> Cart data stored in local storage or server.**
4. **Checkout Process:**
    - **User enters shipping and payment information -> Data submitted for order processing.**

5. Order Placement:

- **Order details saved in Sanity CMS or external database -> Order confirmation displayed.**

6. Shipment Tracking:

- **User requests shipment tracking -> Data fetched from 3rd-party shipping API -> Shipment status displayed.**

7. Post-Purchase Actions:

- **User views past orders (if logged in) -> Order history fetched from Sanity CMS or external database.**

[User]

|

[Product Browsing]

| --> [Sanity CMS] --> [Frontend (Next.js)] --> [ProductCard]

| | (Product Data Displayed)

|

[Cart Addition] --> [Local Storage/Server]

|

[Checkout] --> [User Data] --> [Shipping Info Form] --> [Sanity CMS / Database]

|

[Order Placement] --> [Sanity CMS / External Database]

|

[Shipment Tracking] --> [3rd Party API (Shipping)] --> [Frontend (Tracking Info)]

|

[Post-Purchase] --> [Sanity CMS / Database] --> [Order History Displayed]

Summary of the API Endpoints:

1. /products (GET) – Fetch all product details.
2. /product/{id} (GET) – Fetch details of a specific product.
3. /orders (POST) – Create a new order.
4. /orders/{id} (GET) – Fetch details of a specific order.
5. /shipment/{orderId} (GET) – Track shipment status.
6. /payment (POST) – Process payment for an order.
7. /user/{email} (GET) – Fetch user details and order history.
8. /reviews/{productId} (POST) – Submit a product review.

Sanity CMS Data Schema (for reference):

To implement these endpoints, you will need the following Sanity data schema fields for each product and order:

Product Schema (Example):

- productName
- price
- description
- stock
- image
- category
- rating
- attributes (size, color, etc.)
- relatedProducts

Order Schema (Example):

- customerName
- customerEmail
- shippingAddress
- orderItems (productId, quantity, price)

- **paymentStatus**
- **totalAmount**
- **orderStatus**
- **shipmentId**