**Hackathon Day 2**

1. **Technical Requirements:**

* Frontend Implementation:

User-Friendly Interface: Use React or Next.js to design a clean and intuitive interface. Styling can be handled using tools like Tailwind CSS or Material-UI to ensure a modern look.

Responsive Design: Incorporate media queries and responsive frameworks to make the website function seamlessly on both mobile and desktop devices.

Essential Pages: Create the necessary pages (Home, Product Listing, Product Details, Cart, Checkout, and Order Confirmation) in the pages folder of Next.js.

* Backend with Sanity CMS:

Set up Sanity CMS and define schemas to structure data. For example:

Products Schema: Fields for name, price, description, images, etc.

Customers Schema: Fields for name, email, address, etc.

Orders Schema: Fields for product details, customer info, order status, etc.

Use Sanity Studio to manage content and expose the data through Sanity's APIs for the frontend.

* Third-Party APIs Integration:

Shipment Tracking: Integrate a reliable shipment tracking API, such as Shippo or AfterShip, to keep users updated on their orders.

Payment Gateway: Use secure payment gateway APIs like Stripe or PayPal for seamless transactions.

Ensure proper integration by following API documentation and using HTTP methods (GET, POST, etc.) to fetch and send data between the backend and frontend.

* Testing and Deployment:

Test all features on both mobile and desktop devices to identify and fix any bugs.

Deploy the website using platforms like Vercel or Netlify for efficient hosting and performance.

1. **System Architecture:**

Frontend (Next.js) communicates with Sanity CMS for data.

Sanity CMS connects to the Product Data API for product information and interacts with Third-Party APIs for additional functionalities.

Third-Party APIs handle Shipment Tracking API and Payment Gateway for order and payment processing.

Frontend (Next.js)

User Interacts with frontend to browse products

Sanity CMS

Sanity CMS Fetches Product Data

Shipment & Payment Handled by APIs

Third Party APIs

Product Data API

Shipment Tracking API

Payment Gateway

3.Plan API Requirements:

**1. Products API**

* **Endpoint Name:** /products
* **Method:** GET
* **Description:** Fetch all available products from Sanity CMS.
* **Response Example:**

json

CopyEdit

[

{

"id": 1,

"name": "Product A",

"price": 100,

"stock": 50,

"image": "url-to-product-image"

},

{

"id": 2,

"name": "Product B",

"price": 150,

"stock": 30,

"image": "url-to-product-image"

}

]

**2. Orders API**

* **Endpoint Name:** /orders
* **Method:** POST
* **Description:** Create a new order in Sanity CMS.
* **Payload Example:**

json

CopyEdit

{

"customerInfo": {

"name": "John Doe",

"email": "johndoe@example.com",

"address": "123 Street Name, City, Country"

},

"productDetails": [

{

"productId": 1,

"quantity": 2

},

{

"productId": 2,

"quantity": 1

}

],

"paymentStatus": "Paid"

}

* **Response Example:**

json

CopyEdit

{

"orderId": 456,

"status": "Success",

"message": "Order created successfully"

}

**3. Shipment Tracking API**

* **Endpoint Name:** /shipment
* **Method:** GET
* **Description:** Track order status via a third-party shipment tracking API.
* **Response Example:**

json

CopyEdit

{

"shipmentId": 789,

"orderId": 456,

"status": "In Transit",

"expectedDeliveryDate": "2025-01-20"

}

**4. Express Delivery API (Optional for Q-Commerce):**

* **Endpoint Name:** /express-delivery-status
* **Method:** GET
* **Description:** Fetch real-time delivery updates for perishable items.
* **Response Example:**

json

CopyEdit

{

"orderId": 123,

"status": "In Transit",

"ETA": "15 mins"

}

**5. Rental API (Optional for Rental eCommerce):**

* **Endpoint Name:** /rental-duration
* **Method:** POST
* **Description:** Add rental details for a specific product.
* **Payload Example:**

json

CopyEdit

{

"productId": 456,

"duration": "7 days",

"deposit": 500

}

* **Response Example:**

json

CopyEdit

{

"confirmationId": 789,

"status": "Success",

"message": "Rental created successfully"

}

**Notes for Implementation**

1. **Consistency:** Ensure all endpoints follow RESTful principles.
2. **Authentication:** Secure endpoints with proper authentication (e.g., JWT or OAuth).
3. **Validation:** Validate payloads and responses to prevent errors and ensure data integrity.
4. **Documentation:** Use tools like Swagger or Postman to provide clear and detailed API documentation.

4. Key Points for Technical Documentation:

1. System Architecture Overview

Diagram: Include a flowchart showing interactions between components (Frontend, Sanity CMS, Third-Party APIs).

Components: Briefly explain:

Frontend (Next.js): User interaction layer.

Sanity CMS: Backend for managing products, orders, and customers.

Third-Party APIs: Handles payments, shipment tracking, and other external integrations.

2. Key Workflows

General E-Commerce:

User browses products → Adds to cart → Places order → Tracks shipment.

Q-Commerce:

Real-time inventory updates → Express delivery tracking via /express-delivery-status.

Rental E-Commerce:

Select rental duration → Confirm deposit → Return and condition report.

3. API Endpoints

Endpoint Method Purpose Response Example

/products GET Fetches all product details { "id": 1, "name": "Product A", "price": 100 }

/orders POST Creates a new order { "orderId": 456, "status": "Success" }

/shipment GET Tracks order shipment { "shipmentId": 789, "status": "In Transit" }

/express-delivery-status GET Real-time delivery tracking { "orderId": 123, "ETA": "15 mins" }

4. Sanity Schema Example

Product Schema:

javascript

Copy

Edit

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' }

]

};

5. Technical Roadmap

Milestones:

Week 1: Set up Sanity CMS and define schemas.

Week 2: Develop essential frontend pages and integrate APIs.

Week 3: Test workflows and optimize performance.

Week 4: Finalize and deploy the system.

6. Collaboration Tips

Use GitHub for version control and regular commits.

Brainstorm and share ideas using Slack or Google Meet.

Conduct peer reviews to refine workflows and scalability.