DAY 2: HACKATHON

“TECHNICAL PLANNING OUTCOME”

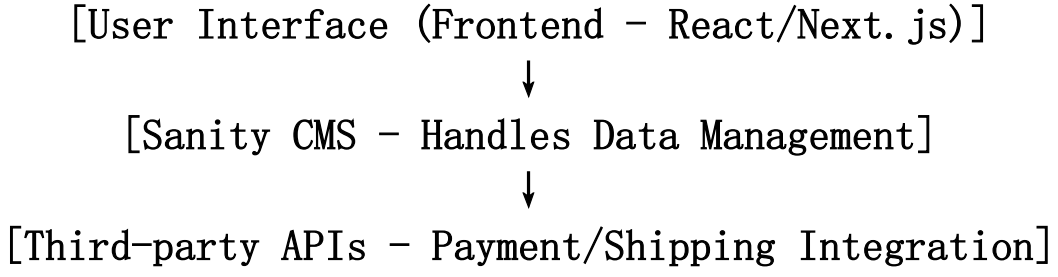
1. System Architecture Instructions:

"System Architecture Overview for Clothing Marketplace"

"This section outlines the high-level system architecture for the Clothing Marketplace. The design includes frontend interaction, content management through Sanity CMS, and third-party integrations for seamless functionality.

Components:

1. **Frontend (React/Next.js)**: Handles user interface for browsing products, managing carts, and placing orders.
2. **Sanity CMS**: Manages product data, customer records, and order processing.
3. **Third-party APIs**: Supports shipment tracking and payment processing.



2. Key Workflows For Clothing Marketplace:

"This section outlines key workflows for the Clothing Marketplace, detailing the steps and interactions between the system's components."

Workflow Example:

1. User browses the marketplace → Request sent to Sanity CMS for product data.
2. Product details displayed dynamically.
3. User places an order → Data sent to Sanity CMS and processed through Payment Gateway API.
4. Confirmation sent to the user → Order tracked via Shipment API.

1. **User Registration:**
    - User fills out the signup form.
    - Data is saved in Sanity CMS.
    - Confirmation email is sent to the user.

User → Frontend (Submit Form) → Sanity CMS (Save Data) → Email Service (Send Confirmation)

2. **Product Browsing:**

    - User selects a category.
    - Sanity API fetches product data.
    - Products are displayed dynamically on the frontend.

User → Frontend (Request Data) → Sanity CMS (Fetch Products) → Frontend (Display Products)

3. **Order Placement:**

   - User adds products to the cart.
   - Order details are saved in Sanity CMS.
   - Payment is processed via Payment Gateway API.
   - Confirmation is shown, and shipment details are tracked.

User → Frontend (Submit Order) → Sanity CMS (Save Order)

→ Payment Gateway (Process Payment) → Sanity CMS (Update Order)

→ Shipment API (Track Order)

3. Plan API Requirements:

Endpoint Name	Methode	Purpose	Request payload	Response
/api/products	GET	Fetch list of products	None	[{ "id": 1, "name": "T-shirt", "price": 20, "size": "M", "color": "Red" }]
/api/products/:id	GET	Fetch product details by ID	None	{ "id": 1, "name": "T-shirt", "price": 20, "size": "M", "color": "Red", "description": "Details" }
/api/orders	POST	Place a new order	{ "userId": 123, "products": [{ "id": 1, "quantity": 2 }], "totalAmount": 40 }	{ "orderId": 456, "status": "Order Placed", "estimatedDelivery": "2025-01-20" }

/api/orders/:id	GET	Fetch order details by ID	None	{ "orderId": 456, "status": "Shipped", "trackingId": "ABC123XYZ" }
/api/auth/register	POST	Register a new user	{ "email": "user@example.com", "password": "securepassword", "name": "John Doe" }	{ "userId": 123, "message": "Registration successful" }
/api/auth/login	POST	User login	{ "email": "user@example.com", "password": "securepassword" }	{ "token": "JWT_TOKEN", "userId": 123, "name": "John Doe" }

4. Sanity Schema Daigram:

Schema: Products:

- Product ID: String
- Name: String
- Price: Number
- Description: String
- Category: String
- Stock: Number
- Image: URL

Schema: Orders:

- Order ID: String
- Customer ID: Reference
- Products: Array of References (Product, Quantity)
- Total Amount: Number
- Order Date: Timestamp
- Status: String

Schema: Customers:

- Customer ID: String
- Name: String
- Email: String
- Phone: String
- Address: String
- Order History: Array of References (Orders)