**Technical Requirements Document**

**1. Technical Requirements**

**Frontend Requirements**

* **Framework**: Next.js for server-side rendering and seamless frontend performance.
* **UI Library**: Material-UI (MUI) for consistent and responsive design components.
* **State Management**: Redux Toolkit for scalable state management.
* **Authentication**: Integration with secure authentication mechanisms (e.g., JWT, OAuth).
* **Image & Video Handling**: Lazy loading and optimized delivery for media content.

**Backend Requirements**

* **Content Management System (CMS)**: Sanity CMS to manage product data, content, and order records.
* **Database**: Managed within Sanity CMS for storing structured product and order data.
* **API Development**: RESTful API integration between the frontend and Sanity CMS.
* **Third-Party APIs**: Integration with external services for shipment tracking and payment processing.

**Third-Party APIs**

* **Shipment Tracking API**: Real-time tracking information integrated via external logistics APIs.
* **Payment Gateway**: Secure payment processing through trusted payment gateways (e.g., Stripe, PayPal).

**2. System Architecture**

**High-Level Architecture Diagram**

**Components**:

* **Frontend (Next.js)**: User interface for browsing products and managing orders.
* **Sanity CMS**: Backend for content and data management.
* **Third-Party APIs**: Shipment tracking and payment processing services.

**Architecture Flow:**

1. **User Browsing Products**:
   * The user accesses the frontend.
   * The frontend requests product data from Sanity CMS via the Product Data API.
   * Product listings and details are dynamically rendered on the frontend.
2. **Order Placement**:
   * The user adds products to the cart and proceeds to checkout.
   * Order details are sent to Sanity CMS via API requests.
3. **Shipment Tracking**:
   * The frontend fetches real-time shipment tracking data through a third-party logistics API.
   * Tracking updates are displayed to the user.
4. **Payment Processing**:
   * Payment details are securely processed through the integrated payment gateway.
   * A confirmation is sent back to the frontend and recorded in Sanity CMS.

**Example System Architecture:**

[Frontend (Next.js)] <---> [Sanity CMS] <---> [Third-Party APIs]

| |

| |

<---> Shipment Tracking API

<--> Payment Gateway

**3. Key Workflows**

**3.1 User Registration**

* **Step 1**: User fills out the registration form.
* **Step 2**: Registration data is sent to Sanity CMS.
* **Step 3**: A confirmation email is sent to the user.

**3.2 Product Browsing**

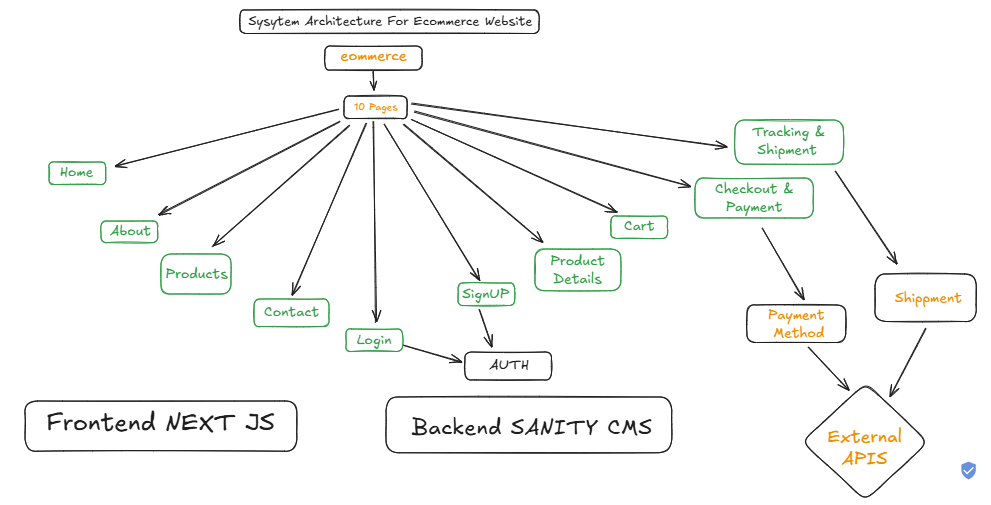
* **Step 1**: User navigates product categories on the frontend.
* **Step 2**: Sanity CMS provides category and product data via API.
* **Step 3**: Products are displayed dynamically on the frontend.

**3.3 Order Placement**

* **Step 1**: User adds items to the cart and proceeds to checkout.
* **Step 2**: Payment details are processed via the payment gateway.
* **Step 3**: Order details are stored in Sanity CMS.
* **Step 4**: Order confirmation is sent to the user.

**3.4 Shipment Tracking**

* **Step 1**: Order status updates are fetched from the third-party logistics API.
* **Step 2**: Shipment status is updated in real-time on the frontend.



**4. Integration Strategy**

**API Integration**

* RESTful API endpoints will be developed for seamless communication between the frontend and backend.
* Third-party APIs will be securely integrated using authentication tokens and encrypted data transfer.

**Security Considerations**

* HTTPS will be enforced for secure communication.
* Sensitive data (e.g., payment details) will be encrypted and handled by secure payment gateways.
* Authentication mechanisms will protect user data and API access.

**Scalability & Performance**

* Lazy loading and image optimization will ensure fast frontend performance.
* Backend services will be optimized for handling large product catalogs and concurrent users.

**5. System Architecture Overview**

**Architecture Diagram**

[Frontend (React/Next.js)] <--> [Backend API (Node.js/Express)] <--> [Sanity CMS] <--> [Third-Party APIs (Shipment)]

**Component Descriptions**

* **Frontend (React/Next.js):** Provides the user interface for product browsing, cart management, and order placement.
* **Backend API (Node.js/Express):** Handles business logic, data processing, and communication between the frontend and Sanity CMS.
* **Sanity CMS:** Stores and manages product, order, and customer data.
* **Third-Party APIs:** Used for shipment tracking and other external services.

**6. Key Workflows**

**Workflow 1: User Browses Products**

1. User accesses the product page on the frontend.
2. Frontend sends a GET request to /products endpoint.
3. Backend fetches product data from Sanity CMS.
4. Frontend displays the product list.

**Workflow 2: User Adds Products to Cart**

1. User clicks "Add to Cart."
2. Frontend updates the cart state and UI.
3. Optionally, cart data can be synced with the backend.

**Workflow 3: User Places an Order**

1. User proceeds to checkout and submits the order.
2. Frontend sends a POST request to /orders with customer and product details.
3. Backend saves the order in Sanity CMS and initiates payment processing.
4. On successful payment, backend confirms the order and triggers shipment processing.

**Workflow 4: User Tracks Order**

1. User checks order status.
2. Frontend sends a GET request to /shipment?order\_id=123.
3. Backend retrieves tracking information from a third-party API.
4. Frontend displays shipment status.

**7. API Endpoints**

| **Endpoint** | **Method** |  | **Purpose** | **Request Payload** | **Response Example** |
| --- | --- | --- | --- | --- | --- |
| /products | GET |  | Fetches all available products from Sanity | None | [ { "id": 1, "name": "Product A", "price": 100, "stock": 20, "image": "url" } ] |
| /orders | POST |  | Creates a new order in Sanity CMS | { "customer": {"name": "John Doe", "email": "john@example.com"}, "products": [{"id": 1, "quantity": 2}], "paymentStatus": "paid" } | { "orderId": "abc123", "status": "confirmed" } |
| /shipment | GET |  | Tracks order shipment via third-party API | ?order\_id=abc123 | { "shipmentId": "xyz789", "status": "In Transit", "expectedDelivery": "2023-09-10" } |

**8. Sanity Schema Example**

**Product Schema**

export default {

name: 'product',

type: 'document',

title: 'Product',

fields: [

{ name: 'name', type: 'string', title: 'Product Name' },

{ name: 'price', type: 'number', title: 'Price' },

{ name: 'stock', type: 'number', title: 'Stock Level' },

{ name: 'image', type: 'image', title: 'Product Image' }

]

};

**Order Schema**

export default {

name: 'order',

type: 'document',

title: 'Order',

fields: [

{ name: 'customerName', type: 'string', title: 'Customer Name' },

{ name: 'customerEmail', type: 'string', title: 'Customer Email' },

{ name: 'products', type: 'array', of: [{ type: 'reference', to: { type: 'product' } }], title: 'Ordered Products' },

{ name: 'paymentStatus', type: 'string', title: 'Payment Status' },

{ name: 'shipmentStatus', type: 'string', title: 'Shipment Status' }

]};

**9. Technical Roadmap**

**Phase 1: Project Setup**

* Initialize frontend with React/Next.js.
* Set up backend with Node.js/Express.
* Configure Sanity CMS.

**Phase 2: API Development**

* Implement /products, /orders, and /shipment endpoints.
* Integrate third-party shipment API.

**Phase 3: Frontend Development**

* Develop product browsing, cart, and checkout UI.
* Implement order placement and tracking features.

**Phase 4: Testing and Deployment**

* Conduct integration testing.
* Deploy application to production.

**10. Conclusion**

This technical requirements document outlines the comprehensive architecture, workflows, API specifications, and Sanity CMS schema design for the eCommerce marketplace platform. By leveraging Sanity CMS for content management, third-party APIs for logistics and payment processing, and a Next.js frontend, the system ensures a seamless, secure, and scalable user experience. This document serves as both a development blueprint and a professional reference for stakeholders.