Marketplace Technical Foundation - General E-Commerce Marketplace

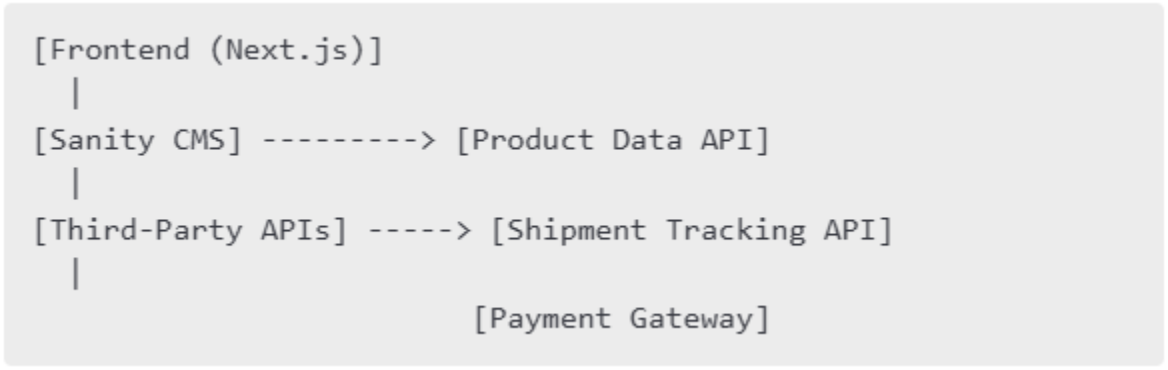
System Architecture Document

Overview: The system architecture for the General E-Commerce Marketplace integrates a responsive frontend with a robust backend managed by Sanity CMS. Third-party APIs handle shipment tracking and payment processing. This modular design ensures scalability and seamless user experiences.

Components:

1. **Frontend (Next.js):**
    - Dynamic and responsive pages for product browsing, cart management, and checkout.
    - Communicates with backend APIs for real-time data updates.
2. **Sanity CMS:**
    - Manages product, customer, and order data.
    - Supports custom schemas for flexible data organization.
3. **Third-Party APIs:**
    - Shipment Tracking API: Provides real-time updates on delivery status.
    - Payment Gateway API: Handles secure payment processing.

Architecture Diagram:



Data Flow:

1. Users interact with the frontend to browse products and place orders.
2. Frontend requests data from the Sanity CMS via API.
3. Orders are recorded in Sanity CMS and updates are fetched from third-party APIs.
4. Real-time shipment and payment updates are displayed to users.

API Specification Document

API Endpoints:

Endpoint	Method	Purpose	Example Payload/Response
/products	GET	Fetch all product details	Response: { "id": 1, "name": "T-shirt", "price": 20 }
/orders	POST	Create a new order	Payload: { "customerId": 123, "products": [...] }
/shipment	GET	Track order shipment status	Response: { "shipmentId": 456, "status": "Delivered" }
/loyalty-points	GET	Fetch loyalty points for a user	Response: { "customerId": 123, "points": 500 }

Example Endpoint Details:

1. **/products**
    - **Method:** GET
    - **Description:** Fetches all available products.
    - **Response:** { "id": 1, "name": "T-shirt", "price": 20, "stock": 50 }
2. **/orders**
    - **Method:** POST
    - **Description:** Creates a new order in the system.
    - **Payload:** { "customerId": 123, "products": [{ "productId": 1, "quantity": 2 }] }
    - **Response:** { "orderId": 789, "status": "Pending" }

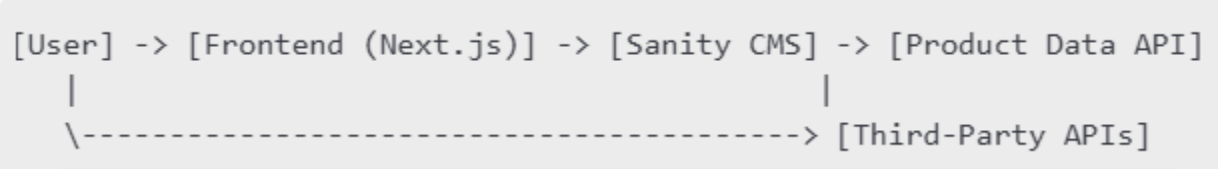
Workflow Diagram

Description: Visual representation of user interactions and data flow within the marketplace.

Workflows:

1. **User Registration:**
    - User signs up -> Data stored in Sanity CMS -> Confirmation email sent.
2. **Product Browsing:**
    - User searches for products -> Sanity CMS provides product data -> Results displayed dynamically.
3. **Order Placement:**
    - User adds items to cart -> Proceeds to checkout -> Order details saved in Sanity CMS -> Payment processed.
4. **Shipment Tracking:**
    - Shipment API fetches delivery status -> Updates displayed on user dashboard.

Diagram:



Data Schema Design

Entities and Attributes:

1. **Products:**
    - Product ID
    - Name
    - Description
    - Price
    - Stock Level
    - Category (e.g., Men, Women, Kids)
    - Sub-Category (e.g., Formal, Casual)
    - Images
    - Tags (e.g., "New Arrival", "Discount")

2. **Orders:**
    - Order ID
    - Customer ID
    - Product List (Product ID, Quantity)
    - Total Amount
    - Status (Pending, Shipped, Delivered)
    - Payment Method
    - Timestamps (Order Date, Expected Delivery Date)

3. **Customers:**
    - Customer ID
    - Name
    - Email
    - Address
    - Contact Number
    - Order History
    - Preferences (Size, Style Preferences)

4. **Shipment:**
    - Shipment ID
    - Order ID
    - Delivery Address
    - Delivery Status
    - Courier Partner
    - Expected Delivery Date

5. **Delivery Zones:**
    - Zone ID
    - Area Name

- Coverage (Cities/Regions)

6. **Loyalty Points:**
    - Points ID
    - Customer ID
    - Points Earned
    - Points Redeemed
    - Expiry Date

Technical Roadmap

Phase 1: Initial Setup

1. Define business goals and target audience.
2. Draft data schema based on marketplace requirements.
3. Set up Sanity CMS for backend management.

Phase 2: System Architecture and API Development

1. Design system architecture and workflows.
2. Develop API endpoints for products, orders, and shipments.
3. Integrate third-party APIs for shipment tracking and payments.

Phase 3: Frontend Development

1. Create responsive pages using Next.js.
2. Implement real-time data fetching from APIs.
3. Develop dynamic features like search filters and personalized recommendations.

Phase 4: Testing and Deployment

1. Conduct unit and integration testing for APIs and frontend components.
2. Deploy the application on a scalable hosting platform.
3. Monitor system performance and resolve any issues.

Phase 5: Post-Launch Optimization

1. Gather user feedback for iterative improvements.
2. Add new features like seasonal collections and advanced loyalty programs.
3. Scale infrastructure to handle increased traffic and transactions.

By adhering to this roadmap, the marketplace will progress systematically from planning to successful deployment.
