# 1.Define Technical Requirements

Based on my business plan, here’s a breakdown of the technical requirements aligned with my e-commerce marketplace:

**Frontend Requirements**

1. **User-Friendly Interface**
   * Intuitive navigation for browsing products.
   * Clear categorization (Home Decor, Appliances, Kitchen Gadgets, etc.).
   * Search functionality with filters (e.g., price range, rating, category).
2. **Responsive Design**
   * Seamless experience across mobile, tablet, and desktop devices.
   * Optimize layout and images for different screen sizes.
3. **Essential Pages**
   * **Home Page**: Highlight featured products, deals, and categories.
   * **Product Listing**: Grid view for browsing all products within a category.
   * **Product Details**: Detailed information, including price, stock, reviews, and dimensions.
   * **Shopping Cart**: Display selected products with quantities and total price.
   * **Checkout**: Form to input billing, shipping information, and payment details.
   * **Order Confirmation**: Summary of the completed purchase, including order ID and tracking info.

**Sanity CMS as Backend**

1. **Product Management**
   * Schema for managing product data:
     + Attributes: Name, Category, Price, Stock, Description, and Images.
   * Integration for updating stock levels automatically post-purchase.
2. **Customer Management**
   * Schema for storing customer data:
     + Attributes: Customer ID, Name, Email, Address, Wishlist, and Order History.
3. **Order Management**
   * Schema for tracking order details:
     + Attributes: Order ID, Products Ordered, Total Price, Payment Status, and Delivery Method.
4. **Design Schema**
   * Align schemas with business goals:
     + Flexibility to add/remove product categories.
     + Support for managing discounts and promotions dynamically.

**Third-Party APIs**

1. **Shipment Tracking**
   * Integration with a logistics API for real-time tracking updates.
   * Display tracking info on the "Order Confirmation" and "My Orders" pages.
2. **Payment Gateways**
   * APIs for secure payment processing (e.g., Stripe, PayPal).
   * Support for multiple payment methods like credit cards and digital wallets.
3. **Other Backend Services**
   * Real-time currency conversion (if applicable).
   * Email/SMS notifications via third-party APIs for order confirmations and updates.

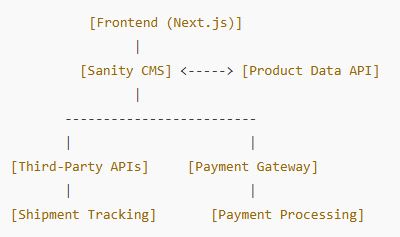
# 2. Design System Architecture

**System Components Overview**

1. **Frontend (Next.js)**
   * Where users interact with the website (browsing, adding products to the cart, placing orders).
2. **Sanity CMS**
   * Acts as the backend, managing products, orders, and user data.
3. **Third-Party APIs**
   * Provides additional features like shipment tracking and email notifications.
4. **Payment Gateway**
   * Handles secure payment transactions and sends payment confirmation.

**System Architecture Diagram**

Here’s a high-level flow of how the components interact:



**Data Flow and Workflows**

**1. User Registration**

* **Steps**:
  + User signs up or logs in on the frontend.
  + User data is sent to Sanity CMS and stored securely.
  + A confirmation email or message is sent via a third-party API.

**2. Product Browsing**

* **Steps**:
  + Users browse categories on the frontend.
  + Frontend sends a request to the **Sanity CMS Product Data API**.
  + The API returns product details, which are displayed on the frontend dynamically.

**3. Order Placement**

* **Steps**:
  + Users add items to the cart and proceed to checkout.
  + Frontend sends order details (products, user info, address) to Sanity CMS via API.
  + Sanity CMS saves the order and updates inventory.

**4. Shipment Tracking**

* **Steps**:
  + Once an order is shipped, shipment tracking info is fetched from a third-party API.
  + The API updates the order status, and this is displayed to the user on the frontend.

**5. Payment Processing**

* **Steps**:
  + Users proceed to pay using the payment gateway.
  + The payment gateway processes the transaction securely.
  + Confirmation of payment is sent to Sanity CMS and the user.