1. Technical Requirements

Frontend Requirements

1. **Framework**: Next.js (React-based, SEO-friendly, fast rendering).
2. **Pages**:
    - **Home Page**: Featured products, banners, and offers.
    - **Product Listing Page**: Filters (category, size, price), sorting.
    - **Product Details Page**: Images, description, size guide, "Add to Cart."
    - **Cart Page**: Display selected products with total price calculation.
    - **Checkout Page**: User details, payment options.
    - **Order Confirmation Page**: Summary, estimated delivery time.
3. **Features**:
    - Responsive design (mobile, tablet, desktop).
    - User authentication (login, signup).
    - Search bar with auto-suggestions.
    - Wishlist and cart functionality.
    - Dark mode (optional).

Backend Requirements

1. **Sanity CMS**:
    - **Schemas**:
        - **Products**: Name, category, price, stock, description, images.
        - **Orders**: User info, products, payment status, order status.
        - **Users**: Authentication, saved addresses, order history.
    - **Benefits**:
        - Easy data management.
        - Real-time updates.
2. **Database**:
    - Sanity CMS acts as the database for structured content.

Third-Party Integrations

1. **Payment Gateway**:
    - Example: Stripe, PayPal.
    - Secure transactions, support for multiple payment methods.
2. **Shipment Tracking API**:
    - Example: FedEx, DHL, or EasyPost.
    - Real-time order tracking.
3. **Email Service**:
    - Example: SendGrid or Mailgun for transactional emails.

4. **Analytics**:
    - Google Analytics for user behavior tracking.

---

2. System Architecture Flow

Workflow:

1. **Frontend**:
    - User browses products → sends request to backend (API).
2. **Backend**:
    - Sanity CMS retrieves product details and sends them to the frontend.
3. **Order Process**:
    - User places order → details saved in Sanity CMS.
    - Payment Gateway processes payment.
4. **Shipment Tracking**:
    - Third-party API fetches order status and updates user

3. API Requirements

Endpoint	Method	Description	Payload	Response
`/products`	GET	Fetch all products	-	`{ id, name, price, stock, image }`
`/products/{id}`	GET	Fetch product details by ID	-	`{ id, name, description, price }`
`/auth/register`	POST	Register a new user	`{ name, email, password }`	`{ userId, status }`
`/auth/login`	POST	User login	`{ email, password }`	`{ token, userId }`
`/orders`	POST	Place a new order	`{ userId, productId, totalPrice }`	`{ orderId, status }`
`/orders/{id}`	GET	Fetch order details by ID	-	`{ orderId, products, status }`
`/shipment/{orderId}`	GET	Track shipment	-	`{ orderId, status, deliveryDate }`

4. Data Schema

Product Schema:

```
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' },
    { name: 'image', type: 'image', title: 'Product Image' },
    { name: 'description', type: 'text', title: 'Description' },
    { name: 'category', type: 'string', title: 'Category' }
  ]
};
```

Order Schema:

```
export default {
  name: 'order',
  type: 'document',
  fields: [
    { name: 'userId', type: 'reference', to: [{ type: 'user' }],
title: 'User' },
    { name: 'productIds', type: 'array', of: [{ type: 'reference', to:
[{ type: 'product' }] }], title: 'Products' },
    { name: 'totalPrice', type: 'number', title: 'Total Price' },
    { name: 'status', type: 'string', title: 'Order Status' }
  ]
};
```

5. Flowchart

Let me create the flowchart for user interaction and system processes.