Day 2: Planning the Technical Foundation

Technical Documentation for General E-Commerce Furniture Marketplace

---

1. System Architecture Overview

High-Level Overview

Our system architecture is designed for a seamless user experience and efficient backend management. The architecture components include:

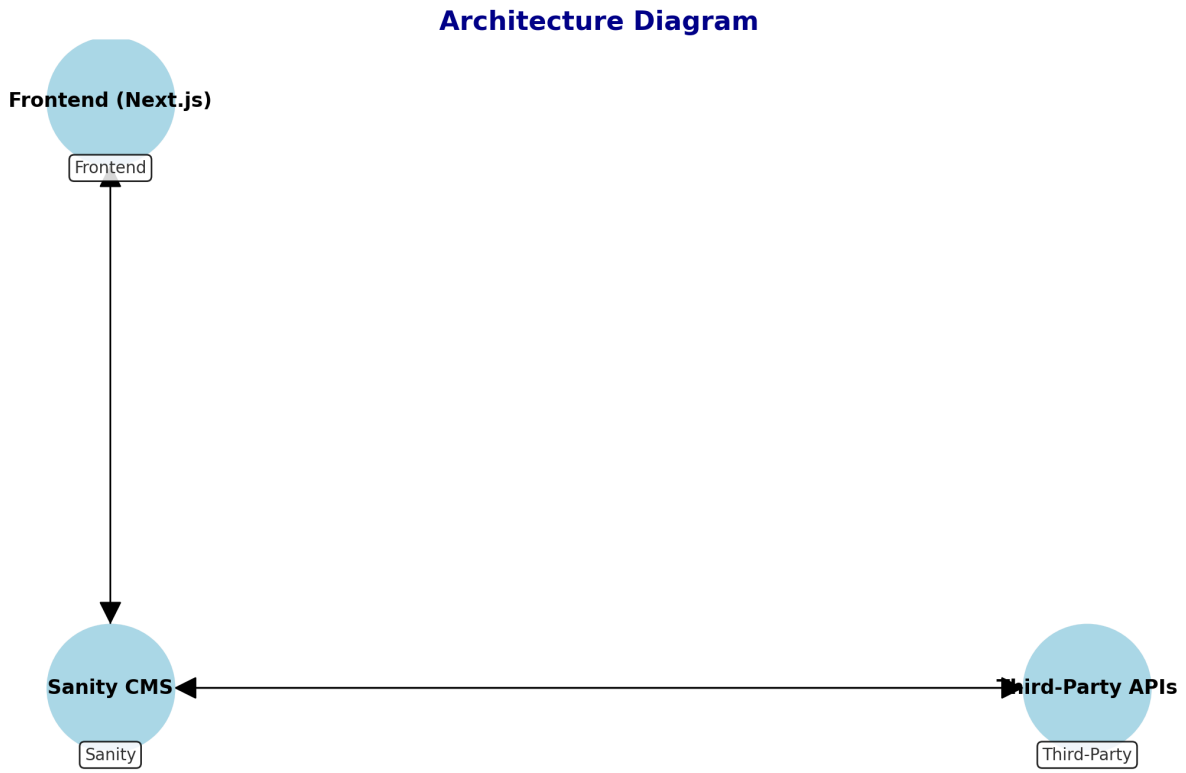
- **Frontend (Next.js/React.js)**:
  Handles user interactions and provides a responsive interface for browsing, cart management, and checkout.

- **Sanity CMS**:
  Serves as the backend database to store and manage:

  - Product information
  - Customer details
  - Orders
- **Third-Party APIs**:
  Integrated for:

  - **Payments** (Stripe, PayPal)
  - **Shipment Tracking**

Architecture Diagram



- **Frontend**: Fetches product data, displays inventory, and interacts with users for checkout and payments.
- **Sanity CMS**: Stores and manages core data such as products, orders, and customers.
- **Third-Party APIs**: Provide real-time tracking and secure payment processing.

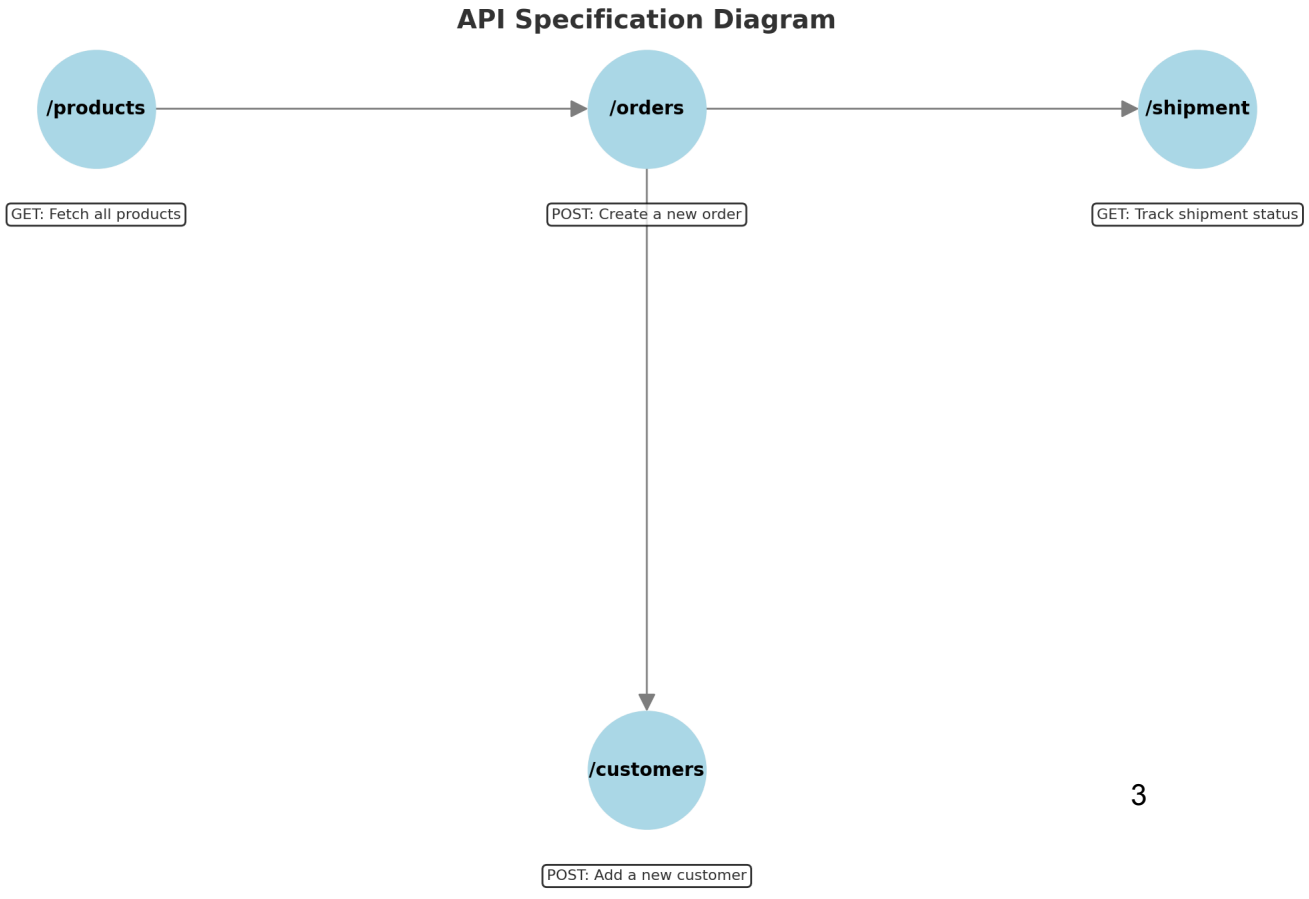
---

2. API Specification

Key Endpoints

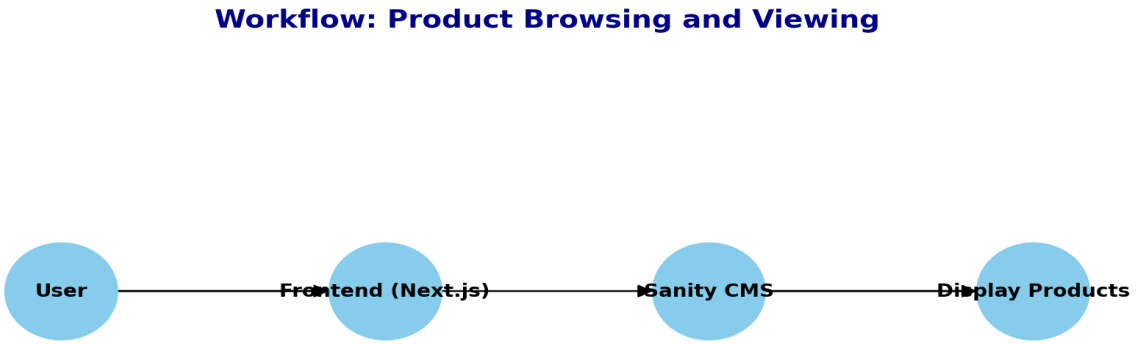
Endpoint	Method	Description	Request/Response Example

`/products`	GET	Fetch all available products	**Response**: `{ "id": 1, "name": "Sofa", "price": 500, "stock": 20, "category": "Living Room" }`
`/orders`	POST	Create a new order	**Request**: `{ "customerID": 1, "products": [ { "id": 1, "qty": 2 } ], "totalAmount": 1000 }` **Response**: `{ "orderID": 123, "status": "Confirmed" }`
`/customers`	POST	Add a new customer	**Request**: `{ "name": "John Doe", "email": "john@example.com", "phone": "1234567890", "address": "123 Main St" }` **Response**: `{ "customerID": 1, "status": "Success" }`
`/shipment`	GET	Track shipment status	**Response**: `{ "shipmentID": 456, "status": "In Transit", "ETA": "2 days" }`

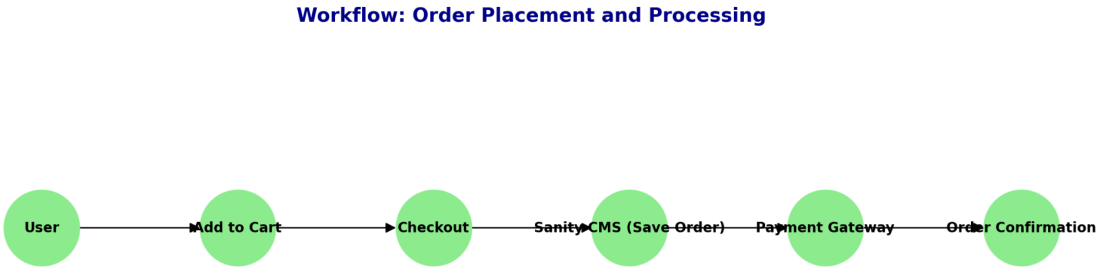


3. Workflow Diagrams

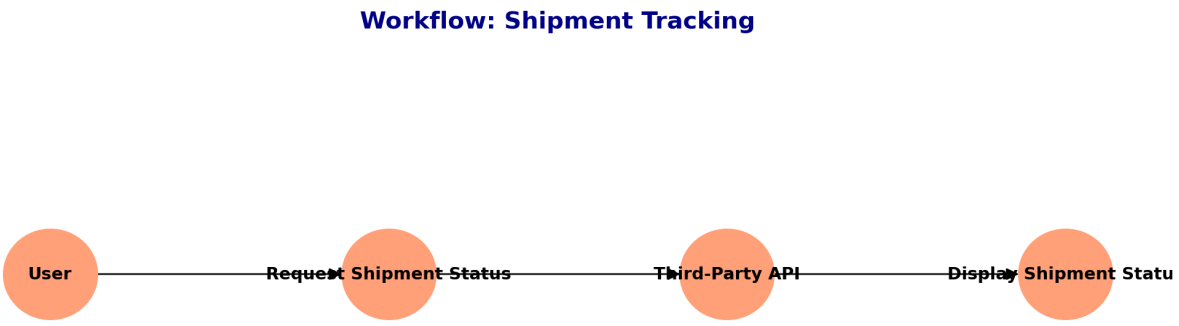
1. Product Browsing and Viewing



2. Order Placement and Processing



3. Shipment Tracking



---

4. Sanity Schema

Products 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 Quantity' },
    { name: 'category', type: 'string', title: 'Category' },
    { name: 'imageURL', type: 'url', title: 'Image URL' }
  ]
};
```

Orders Schema

```
export default {
  name: 'order',
  type: 'document',
  fields: [
    { name: 'orderID', type: 'number', title: 'Order ID' },
    { name: 'customerID', type: 'reference', to: [{ type: 'customer' }], title: 'Customer' },
    { name: 'products', type: 'array', of: [{ type: 'reference', to: [{ type: 'product' }] }],
title: 'Products' },
    { name: 'totalAmount', type: 'number', title: 'Total Amount' },
    { name: 'status', type: 'string', title: 'Order Status' }
  ]
};
```

Customers Schema

```
export default {
  name: 'customer',
  type: 'document',
  fields: [
    { name: 'name', type: 'string', title: 'Customer Name' },
    { name: 'email', type: 'string', title: 'Email' },
    { name: 'phone', type: 'string', title: 'Phone Number' },
    { name: 'address', type: 'string', title: 'Address' }
  ]
};
```

---

5. Outcomes and Key Takeaways

By completing the technical foundation, the marketplace now has:

1. A **clear system architecture** that aligns with business goals.
2. Defined **API endpoints** for core functionality (products, orders, customers, and shipments).
3. Visualized **workflows** for browsing, ordering, and tracking.
4. Ready-to-use **Sanity CMS schemas** for efficient backend data management.

This documentation forms the backbone of my marketplace, ensuring a smooth transition to implementation. Let me know if you need any further refinements or assistance!