FURNIRO.COm

TECHNICAL DOCUMENTATION FOR FURNITURE MARKETPLACE

1. Introduction

1.1 Project Overview

This project is a Furniture E-Commerce Marketplace that offers a wide range of furniture products. It caters to users searching for high-quality and customizable furniture. The platform ensures a seamless user experience with responsive design and a robust backend for efficient management.

1.2 Purpose of the Document

This document serves as a comprehensive guide for the technical development of the Furniture Marketplace. It defines system architecture, workflows, API integrations, and data schemas that align the platform to be scalable, user-friendly, and meet business objectives.

2. System Architecture

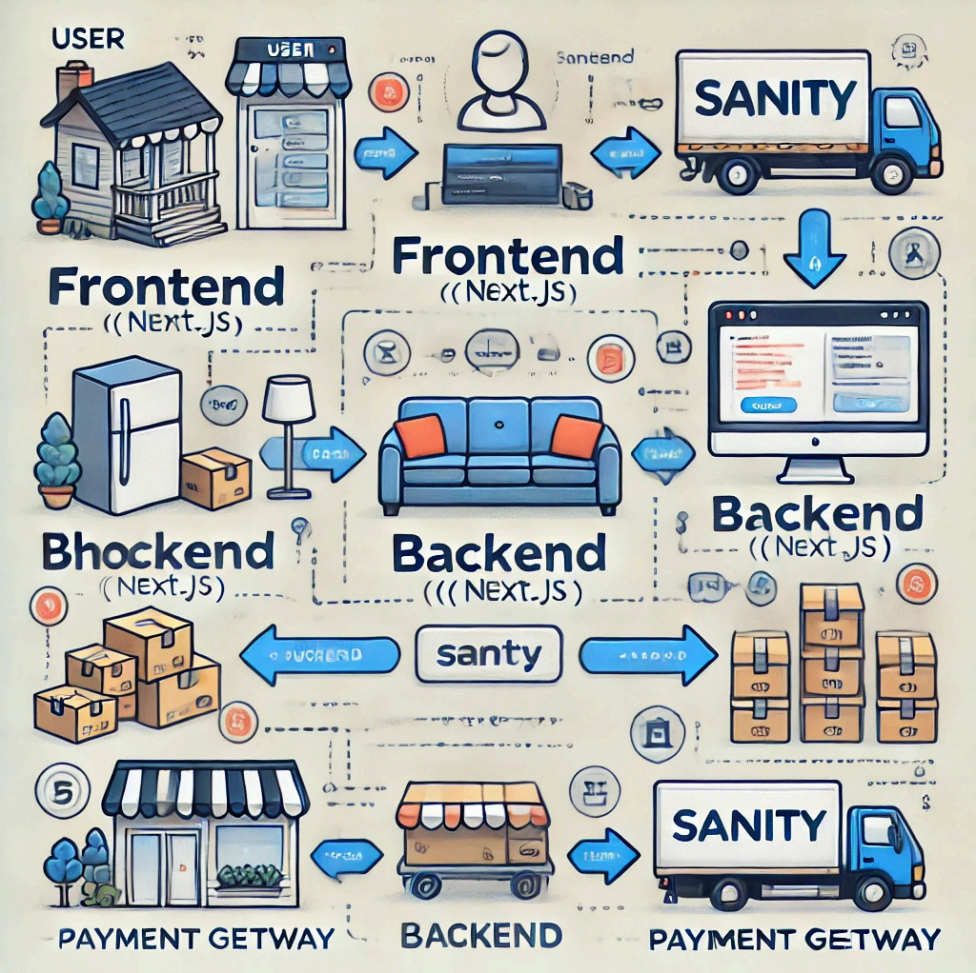
2.1 High-Level Architecture

- Frontend:
  - Next.js is used for dynamic, server-side rendering and React-based components.
  - Tailwind CSS is utilized for implementing responsive and modern UI design.
- Authentication:
  - User login and registration are handled via Clerk API, ensuring secure and seamless authentication.
- Backend:
  - Sanity CMS is used for managing:

    - Product catalog (name, price, descriptions, images, stock levels).
    - User profiles and order histories.
    - Customization options for furniture products.
- Third-Party Integrations:
    - Payment Gateway: Secure transactions are handled via Stripe API.
    - Shipping Services: Real-time shipment tracking is managed through ShipEngine API.

2.2 Architecture Diagram

Example Diagram:



3. Technical Requirements

3.1 Frontend Requirements

- Pages to include:
    - Home Page: Showcasing featured products and categories.
    - Category Page: Filtering furniture types (sofas, beds, chairs, etc.).
    - Product Detail Page: Displaying detailed information and customization options.
    - Cart Page: Showing selected products and total price.
    - Order Tracking Page: Shipment status updates.
- Responsive design for mobile and desktop devices.

3.2 Backend Requirements

- **Sanity CMS configuration:**
  - **Managing product listings.**
  - **Tracking orders and inventory.**
  - **Storing customer data.**
- **Real-time updates for stock levels and order statuses.**

3.3 API Integrations

- **Clerk API: For user authentication and profile management.**
- **Stripe API: To handle secure payments.**
- **ShipEngine API: For tracking shipments and calculating delivery times.**

4. Workflows

4.1 User Registration

- **Users register or log in via the frontend.**
- **Clerk API handles authentication.**
- **User details are securely stored in Sanity CMS.**

4.2 Product Browsing

- **Users can explore product categories and apply filters (price, type, customization).**
- **Product data is fetched dynamically from Sanity CMS.**

4.3 Order Placement

- **Users add items to their cart and proceed to checkout.**
- **Stripe API processes payments securely.**
- **Order details are saved in Sanity CMS, and users receive confirmation.**

4.4 Shipment Tracking

- **ShipEngine API fetches shipment statuses.**
- **Real-time updates are displayed on the Order Tracking Page.**

5. Data Schema Design

5.1 Product Schema (Sanity CMS)

```
export default {
    name: 'product',
    type: 'document',
    fields: [
      { name: 'name', type: 'string', title: 'Product Name' },
      { name: 'slug', type: 'slug', title: 'Slug', options: { source: 'name' } },
      { name: 'description', type: 'text', title: 'Description' },
      { name: 'price', type: 'number', title: 'Price' },
      { name: 'stock', type: 'number', title: 'Stock Level' },
      { name: 'images', type: 'array', of: [{ type: 'image' }], title: 'Product Images' },
      { name: 'customizations', type: 'array', of: [{ type: 'string' }], title: 'Customizations' },
    ],
  };
```

5.2 Order Schema

```


export default {
    name: 'order',
    type: 'document',
    fields: [
      { name: 'productId', type: 'reference', to: [{ type: 'product' }], title: 'Product ID' },
      { name: 'price', type: 'number', title: 'Price' },
      { name: 'quantity', type: 'number', title: 'Quantity' },
      { name: 'subtotal', type: 'number', title: 'Subtotal' },
    ],
  };


```

5.3 Customer Schema

```

export default {
    name: 'customer',
    type: 'document',
    fields: [
      { name: 'name', type: 'string', title: 'Customer Name' },
      { name: 'company', type: 'string', title: 'Company Name' },
      { name: 'country', type: 'string', title: 'Country' },
      { name: 'address', type: 'string', title: 'Address' },
      { name: 'city', type: 'string', title: 'City' },
      { name: 'province', type: 'string', title: 'Province' },
      { name: 'phone', type: 'string', title: 'Phone Number' },
      { name: 'email', type: 'string', title: 'Email Address' },
      { name: 'additionalInfo', type: 'text', title: 'Additional Info' },
    ],
  };


```

5.4 Contact as Company Schema

```
export default {
    name: 'contact',
    type: 'document',
    fields: [
      { name: 'name', type: 'string', title: 'Name' },
      { name: 'email', type: 'string', title: 'Email' },
      { name: 'subject', type: 'string', title: 'Subject' },
      { name: 'message', type: 'text', title: 'Message' },
    ],
  };
```

5.5 Delivery Zone Schema

```
export default {
    name: 'deliveryZone',
    type: 'document',
    fields: [
      { name: 'zoneName', type: 'string', title: 'Zone Name' },
      { name: 'regionsCovered', type: 'array', of: [{ type: 'string' }], title: 'Regions Covered' },
      { name: 'deliveryTime', type: 'string', title: 'Delivery Time' },
      { name: 'cost', type: 'number', title: 'Cost' },
      { name: 'specialNotes', type: 'text', title: 'Special Notes' },
    ],
  };
```

5.6 ShipEngine Schema

```
export default {
    name: 'shipEngine',
    type: 'document',
    fields: [
      { name: 'shipmentId', type: 'string', title: 'Shipment ID' },
      { name: 'carrier', type: 'string', title: 'Carrier' },
      { name: 'trackingNumber', type: 'string', title: 'Tracking Number' },
      { name: 'deliveryStatus', type: 'string', title: 'Delivery Status' },
      { name: 'estimatedDelivery', type: 'datetime', title: 'Estimated Delivery' },
    ],
  };
```

6. Technical Roadmap

Phase 1: Frontend Development

- **Create UI components using Next.js and Tailwind CSS.**
- **Implement responsive design for all devices.**

Phase 2: Backend Setup

- **Configure Sanity CMS for managing product and order data.**
- **Define schemas for products, orders, and customers.**

Phase 3: API Integration

- **Integrate Clerk API for user authentication.**
- **Set up Stripe API for secure payment processing.**
- **Configure ShipEngine API for shipment tracking.**

--- Prepared by ROHMA SHABBIR ---