Technical Foundation for the Furniture Marketplace

1.Introduction

The furniture marketplace is designed to offer a seamless platform where customers can explore, customize, and purchase premium furniture items. The platform focuses on a user-friendly interface, dynamic product management, and secure payment processing to ensure a smooth customer experience. The marketplace will feature modern design principles and responsive layouts to cater to a diverse audience, providing functionality such as browsing products, managing orders, and tracking deliveries.

The backend is powered by Sanity CMS for content management, enabling efficient product and order management. Next.js powers the frontend, ensuring fast page loading, SEO optimization, and scalability. The ultimate goal is to create a reliable, scalable, and user-focused platform to support business growth.

2.System Architecture

Architecture Overview

1.Frontend:

Built using Next.js, leveraging server-side rendering (SSR) for improved performance and SEO.

Styling is implemented using Tailwind CSS, ensuring a consistent and responsive design system.

2.Backend:

Utilizes a Node.js environment for handling API requests and processing business logic.

Integrates with Sanity CMS for seamless content and data management.

3.Database:

Sanity CMS acts as the backend database for storing product details, customer orders, and related metadata.

GROQ (Graph-Relational Object Queries) is used to fetch and manage content efficiently.

4.Third-Party Services:

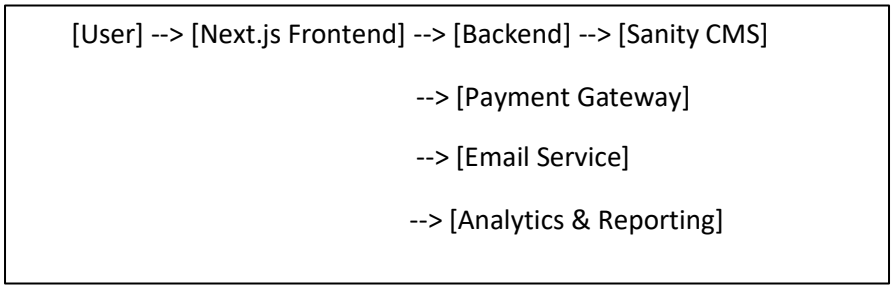
Payment Gateway API (e.g., Stripe): Ensures secure and fast transactions for customers.

Email Service (e.g., SendGrid): Handles transactional email notifications for orders.

5.Deployment:

Hosted on Vercel for streamlined deployment and scalability.

Architecture Diagram



This modular architecture ensures that each component can be maintained and scaled independently.

3.Key Workflows

Browsing Products

1.Customers land on the homepage, which highlights featured furniture items.

2.They navigate to the product catalog via categories like "Living Room," "Bedroom," or "Office."

3.Filtering and sorting options enable users to refine their search based on price, style, and availability.

4.Clicking on a product directs the user to a product detail page, showing specifications, multiple images, and customer reviews.

Adding to Cart

1.Users can add products to the cart from the product detail page.

2.A dynamic cart dropdown provides a quick overview of selected items, prices, and quantities.

3.Items remain in the cart for logged-in users, even if they navigate away or log out.

Placing an Order

1.At checkout, users provide shipping details and select a payment method.

2.Payment details are securely handled through the integrated payment gateway.

3.Once payment is processed, an order confirmation page is displayed, and the order data is stored in Sanity CMS.

Tracking Shipments:

1.After placing an order, customers receive a confirmation email with an order ID.

2.They can use this ID to track the order's status (e.g., "Processing," "Shipped," "Delivered") via their account dashboard.

4.API Specifications

Products API

GET /api/products

Description: Fetches a list of all available products.

Response:

[

{ "id": "1", "title": "Sofa Set", "price": 1000, "image": "/sofa.jpg" },

{ "id": "2", "title": "Dining Table", "price": 2000, "image": "/table.jpg" }

]

GET /api/products/:id

Description: Fetches details of a specific product.

Response:

{ "id": "1", "title": "Sofa Set", "price": 1000, "description": "Luxury sofa set.", "image": "/sofa.jpg" }

Orders API:

POST /api/orders

Description: Creates a new order with the user's cart and payment details.

Payload:

{ "userId": "123", "cart": [{ "productId": "1", "quantity": 1 }] }

GET /api/orders/:userId

Description: Retrieves all orders for a specific user.

5.Sanity CMS Schema Design

Product Schema

Stores product information such as name, price, description, and images.

```
 export default {
 name: "product",
 type: "document",
 title: "Product",
 fields: [
  { name: "title", type: "string", title: "Product Name" },
  { name: "description", type: "text", title: "Description" },
  { name: "price", type: "number", title: "Price" },
  { name: "image", type: "image", title: "Image" },
  { name: "category", type: "string", title: "Category" },
 ],
};
```

Order Schema

Manages order-related details for customers.

```
 export default {
```

```
  name: "order",
  type: "document",
  title: "Order",
  fields: [
    { name: "userId", type: "string", title: "User ID" },
    { name: "products", type: "array", of: [{ type: "reference", to: [{ type: "product" }] }] },
    { name: "status", type: "string", title: "Order Status" },
  ],
};
```

6.Frontend Details

Pages Structure:

Homepage (/): Displays featured products and promotional banners.

Catalog (/products): Lists all available products with filtering options.

Product Details (/products/[id]): Detailed page for a specific product.

Cart (/cart): Displays selected products for checkout.

Order Success (/order-success): Confirms order placement.

Design Principles:

Responsive design ensures mobile and desktop compatibility.

Consistent use of Tailwind CSS utility class for styling.

7. A flowchart representing the whole content: