Marketplace Technical Foundation

FurniStore

---

1. System Architecture Document

1.1 Overview

The marketplace website for FurniStore is designed to allow customers to browse and purchase premium furniture. The system consists of several key components:

- **Frontend**: A React-based web application that provides the user interface for customers to interact with the marketplace.
- **Backend**: A mock API that simulates the processing of data, including fetching furniture items, handling user interactions, and performing CRUD operations.
- **CMS**: Sanity CMS to manage and store content such as furniture details, categories, and other relevant information.
- **Database**: The database is integrated with Sanity to store data related to furniture items, customer profiles, and order histories.

---

1. Frontend

Key Pages:

- **Homepage**: A page showcasing banners, featured furniture, categories, and easy navigation.
- **Furniture Listing**: Displays a grid of furniture items with filters such as material, type, category, and price range.
- **Furniture Details**: A page showing detailed information for each furniture item (description, dimensions, material, images).
- **Cart**: A section to review and modify selected items before checkout.
- **Checkout**: A user-friendly interface for finalizing orders, entering payment information, and selecting delivery options.
- **Order Tracking**: A dedicated page to track the delivery status of placed orders.

---

2. Sanity CMS

Key Functionalities:

- **Product Management**: Store and manage furniture details like name, price, category, material, dimensions, images, and descriptions.
- **Order Management**: Track customer orders, payment status, and shipping details.

---

3. Third-Party APIs

1. Payment Gateway:

- Integrate Stripe or PayPal for secure and reliable payment processing.
- Features include:
  - Credit/debit card payment support.
  - Multi-currency support for international transactions.
  - Refund and dispute management.

2. Shipment Tracking:

- Use AfterShip or similar courier APIs to provide real-time tracking information for customer orders.
- Features include:
  - Real-time status updates (e.g., Shipped, Out for Delivery, Delivered).
  - Notifications for order movement.

3. Mock API:

- Create a Mock API for development and testing purposes to simulate data and API responses without relying on live APIs.
- Features include:
  - Simulating furniture data, order statuses, and tracking updates.

---

4. API Structure

Products API

- `/api/products` (GET): Fetch all furniture products.
- `/api/products/{id}` (GET): Fetch a single furniture product by ID.
- `/api/products` (POST): Add a new furniture product.
- `/api/products/{id}` (PUT): Update furniture product details.
- `/api/products/{id}` (DELETE): Delete a furniture product.

Categories API

- `/api/categories` (GET): Fetch all furniture categories.

- `/api/categories/{id}` (GET): Fetch a single category by ID.

Orders API

- `/api/orders` (GET): Fetch all orders for a user.
- `/api/orders/{id}` (GET): Fetch details of a single order.
- `/api/orders` (POST): Place a new order.
- `/api/orders/{id}/ship` (POST): Mark an order as shipped.
- `/api/orders/{id}/tracking` (GET): Fetch tracking information for a shipped order.

Users API

- `/api/users/{id}` (GET): Fetch user profile details.

Authentication API

- `/api/auth/register` (POST): User registration.
- `/api/auth/login` (POST): User login.

Cart API

- `/api/cart` (GET): Fetch cart details for a user.
- `/api/cart/{id}` (POST): Add an item to the user's cart.
- `/api/cart/{id}` (DELETE): Remove an item from the user's cart.

---

5. System Workflow

Login/Signup

- **User Action**: User initiates login or signup.
- **API Call**: `POST /api/auth/register`.
- **Data Management**: User details are saved in Sanity.

Home Page

- **User Action**: User lands on the homepage after login or signup.

Product Page

- **User Action**: User views a list of available furniture.
- **API Call**: `GET /api/products`.
- **Data Management**: Backend fetches all product data from Sanity.

Single Product Page

- **User Action**: User clicks on a product to view details.
- **API Call**: `GET /api/products/{id}`.
- **Data Management**: Backend fetches the product details by ID from Sanity.

Add to Cart

- **User Action**: User adds a product to their cart.
- **API Call**:
  - `GET /api/cart` to fetch current cart details.
  - `POST /api/cart/{id}` to add the product to the cart.
- **Data Management**: Backend updates the user's cart details in Sanity.

Checkout

- **User Action**: User finalizes the order.
- **API Calls**:
  - `POST /api/orders` to create a new order.
  - `POST /api/cart/{id}` to confirm items in the cart.
- **Data Management**: Backend saves the order details and updates the cart in Sanity.

Track Order

- **User Action**: User tracks their order status.
- **API Call**: `GET /api/orders/{id}/tracking`.
- **Data Management**: Backend fetches tracking details and provides them to the user.

Sanity Schema

Furniture Schema

```
javascript
CopyEdit
export default {
  name: 'furniture',
  title: 'Furniture',
  type: 'document',
  fields: [
    { name: 'name', title: 'Furniture Name', type: 'string', validation: Rule
=> Rule.required().min(3).max(50) },
    { name: 'slug', title: 'Slug', type: 'slug', options: { source: 'name',
maxLength: 96 }, validation: Rule => Rule.required() },
    { name: 'description', title: 'Description', type: 'text', validation:
Rule => Rule.required().max(300) },
    { name: 'price', title: 'Price', type: 'number', validation: Rule =>
Rule.required().positive() },
    { name: 'category', title: 'Category', type: 'string', options: { list:
[{ title: 'Sofas', value: 'sofas' }, { title: 'Beds', value: 'beds' }, {
```

```
title: 'Chairs', value: 'chairs' }, { title: 'Tables', value: 'tables' }] },
validation: Rule => Rule.required() },
    { name: 'material', title: 'Material', type: 'string', validation: Rule
=> Rule.required() },
    { name: 'dimensions', title: 'Dimensions', type: 'string', validation:
Rule => Rule.required() },
    { name: 'images', title: 'Images', type: 'array', of: [{ type: 'image'
}], options: { hotspot: true } },
    { name: 'stock', title: 'Stock', type: 'number', validation: Rule =>
Rule.required().min(0) },
    { name: 'createdAt', title: 'Created At', type: 'datetime', initialValue:
() => new Date().toISOString() },
  ],
};
```
