"Marketplace Technical Foundation - Chairify"

Chairify - Technical Foundation Report

1. Define Technical Requirements

Frontend Requirements

- **User Interface**:
    - Responsive design optimized for mobile and desktop devices.
    - Essential pages:
        1. Home Page: Highlights featured chairs, categories, and promotions.
        2. Product Listing Page: Displays chairs with filters (e.g., type, price, and material).
        3. Product Details Page: Includes product images, descriptions, reviews, and stock availability.
        4. Cart Page: Displays selected items with options to update quantities.
        5. Checkout Page: Handles payment and user details.
        6. Order Confirmation Page: Displays the summary of the completed order.
- **Usability Features**:
    - Search bar with auto-suggestions.
    - Sorting and filtering options for products.

Backend Requirements (Sanity CMS)

- Manage the following:
    - Product data (name, price, stock, image, material, dimensions).
    - Customer data (name, email, address).
    - Order records (order ID, products purchased, order status).
- Design schemas in Sanity CMS for:
    - Products.
    - Orders.
    - Users.

Third-Party API Integrations

- **Shipment Tracking API**:
    - Real-time tracking of shipments.
    - Delivery updates for customers.
- **Payment Gateway API**:
    - Secure payment handling for credit/debit cards and wallets.

2. Design System Architecture

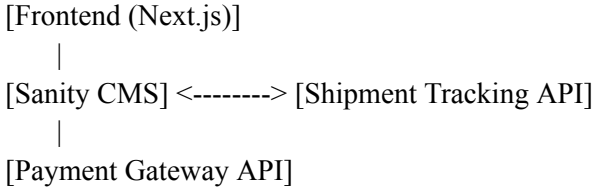
System Components Overview

1. **Frontend**: Built with Next.js for fast, dynamic, and SEO-friendly pages.
2. **Sanity CMS**: Manages all data entities such as products, orders, and customers.
3. **Third-Party APIs**: Handles logistics and payment processing.

Data Flow

1. User browses the marketplace.
2. The frontend fetches product data from Sanity CMS through a RESTful API.
3. On order placement:
    - Order details are sent to Sanity CMS.
    - Payment Gateway API processes the transaction.
    - Shipment Tracking API updates order status.

System Architecture Diagram



3. Key Workflows

1. User Registration

- User signs up with an email and password.
- Data is stored in Sanity CMS.

2. Product Browsing

- User visits the product listing page.
- Frontend fetches product data (name, price, image, stock) from Sanity CMS.

3. Order Placement

- User adds items to the cart.
- During checkout:
    - Order details are saved in Sanity CMS.
    - Payment is processed through the Payment Gateway API.
    - Shipment details are fetched from the Shipment Tracking API.

4. Shipment Tracking

- Users can view real-time shipment status.

4. API Requirements

Endpoint	Method	Description	Response
/products	GET	Fetch all products from Sanity CMS.	{ "id": 1, "name": "Chair A", "price": 100 }
/product/{id}	GET	Fetch product details by ID.	{ "id": 1, "name": "Chair A", "description": "..." }
/cart	POST	Add items to the cart.	{ "cartId": 123, "status": "Added" }
/orders	POST	Save new order details in Sanity CMS.	{ "orderId": 456, "status": "Success" }
/shipment/{id}	GET	Fetch real-time shipment tracking details.	{ "status": "In Transit", "ETA": "2 days" }

5. Sanity Schema Design

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: 'material', type: 'string', title: 'Material' },
    { name: 'dimensions', type: 'string', title: 'Dimensions' },
    { name: 'image', type: 'image', title: 'Image' }
  ]
};
```

Order Schema

```
export default {
  name: 'order',
  type: 'document',
  fields: [
    { name: 'customerId', type: 'string', title: 'Customer ID' },
    { name: 'products', type: 'array', of: [{ type: 'reference', to: { type: 'product' } }], title: 'Products' },
    { name: 'totalPrice', type: 'number', title: 'Total Price' },
```

```
    { name: 'status', type: 'string', title: 'Order Status' }
  ]
};
```

6. Documentation and Collaboration

Technical Documentation

1. **System Architecture**: Includes diagrams and detailed descriptions of component interactions.
2. **API Specifications**: Lists endpoints, methods, and expected responses.
3. **Sanity Schemas**: Outlines how data is structured.

Collaboration Guidelines

- Use GitHub for version control.
- Share workflows and API designs with peers for feedback.
- Regular brainstorming sessions to refine features.

7. Technical Requirements for Sanity CMS as the Backend

Sanity CMS will serve as the backend for the marketplace, acting as a headless content management system to manage and deliver structured data. This provides a scalable and flexible infrastructure to meet the marketplace's operational needs.

1. Core Functions of Sanity CMS

- **Product Data Management**:
    - Store and manage product information such as names, descriptions, prices, stock levels, categories, and images.
    - Allow dynamic updates to product listings without requiring frontend redeployment.
- **Customer Data Management**:
    - Maintain customer details, including account information, shipping addresses, and order histories.
    - Enable personalized user experiences and targeted interactions.
- **Order Records**:
    - Store details about orders, including product information, customer data, payment statuses, and timestamps.
    - Support seamless order tracking and efficient management.

2. Sanity CMS as the Database

- **Structured Content**:
    - Define custom schemas for key data types such as products, customers, and orders to match specific business requirements.
- **Real-Time Updates**:
    - Utilize Sanity’s real-time APIs to instantly reflect backend changes on the frontend.
- **Scalability**:

    - Flexible schema definitions enable future enhancements, such as adding fields for promotions, reviews, or shipment tracking.

3. Key Schemas

- **Product Schema**:
    - Fields include: name, price, description, image, category, stock, and rating.
- **Customer Schema**:
    - Fields include: name, email, password, address, and orderHistory.
- **Order Schema**:
    - Fields include: orderID, customerID, products, totalAmount, status, and orderDate.

4. Integration with Frontend

- **API Queries**:
    - Use **GROQ** (Graph-Relational Object Queries) to efficiently retrieve data from Sanity CMS.
    - Example queries:
        - Fetch all products: *[_type == "product"]
        - Fetch orders for a specific customer: *[_type == "order" && customerID == $id]
- **Data Flow**:
    - Sanity CMS acts as the single source of truth, dynamically providing content for the frontend through Next.js routes.

5. Advantages of Using Sanity CMS

- **Customizable**:
    - Create tailored schemas to suit the marketplace's specific needs.
- **User-Friendly Content Studio**:
    - Offers an intuitive interface, making it easy for non-technical users to update data.
- **Real-Time Collaboration**:
    - Allows multiple team members to make updates simultaneously.
- **Built-In APIs**:
    - Provides powerful, ready-to-use APIs for CRUD operations, simplifying backend development.

Key Takeaways

By the end of this process, **Chairify** will have:

- A responsive, user-friendly marketplace.
- Sanity CMS-powered data management.
- Secure and scalable API integrations.
- Comprehensive technical documentation for seamless development.