MARKETPLACE TECHNICAL FOUNDATION - FURNITURE BAZAAR

Table of Contents

---

1. System Architecture Overview

High-Level Architecture Diagram

The architecture integrates three main components:

1. Frontend (Next.js with shadcn):

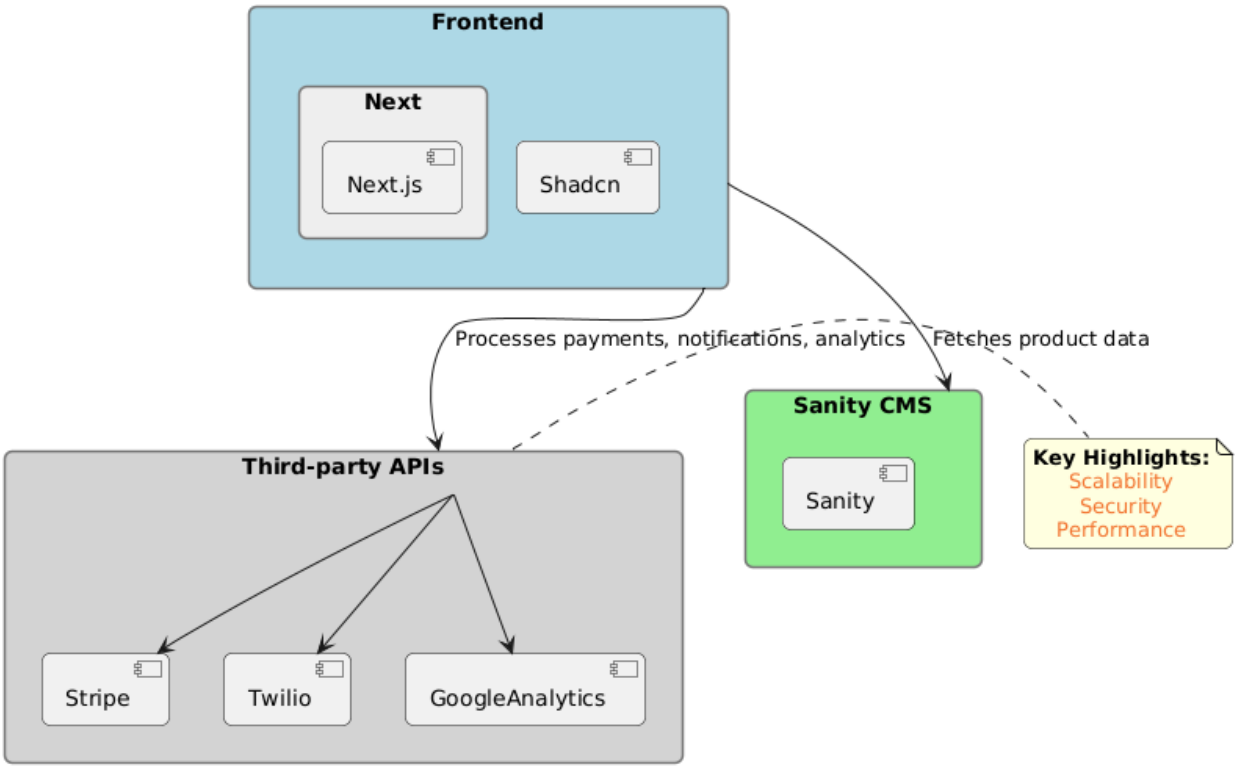
- Manages user interface and interactions, ensuring a seamless browsing and shopping experience.
- Handles real-time data updates and page responsiveness.

2. Sanity CMS:

- Acts as the primary data source for product information, categories, and orders.
- Enables efficient content management and updates without redeployment.

3. **Third-party APIs:**

- **Payment Gateway (e.g., Stripe):** Processes secure transactions.
- **Notification Service (e.g., Twilio):** Sends order confirmations via SMS or email.
- **Analytics Tools (e.g., Google Analytics):** Monitors user behavior and optimizes the user journey.



Key Highlights:

- **Scalability:** Modular design allows easy addition of features like user reviews or loyalty programs.
- **Security:** Data protection through HTTPS, secure tokens, and encrypted communications.
- **Performance:** Optimized for fast load times and minimal latency.

---

2. Key Workflows

User Registration

Objective: Provide a simple and secure way for users to create accounts and log in.

1. **Frontend:**
    - Displays interactive registration and login forms with real-time validation.
    - Integrates social login options (e.g., Google, Facebook).
2. **Backend API:**
    - **POST /api/auth/register:** Handles user registration.
    - **POST /api/auth/login:** Authenticates users and issues secure tokens.
3. **Sanity CMS:**
    - Stores optional user preferences (e.g., wishlist items, browsing history).

Product Browsing

Objective: Ensure users can explore products effortlessly.

1. **Frontend:**
    - Showcases categorized furniture (e.g., Sofas, Beds, Tables, Chairs).
    - Implements dynamic search and filter capabilities.
    - Supports wishlist functionality.
2. **Backend API:**
    - **GET /api/products:** Retrieves filtered product listings.
    - **GET /api/products/:id:** Fetches detailed product information.
3. **Sanity CMS:**
    - Serves optimized product data (name, price, images, availability).

Order Placement

Objective: Enable smooth and secure order creation and processing.

1. **Frontend:**
    - Displays a detailed checkout flow, including cart review and payment options.
    - Offers multiple payment methods and auto-applies available discounts.
2. **Backend API:**
    - **POST /api/orders:** Creates and validates orders.
3. **Third-party APIs:**
    - **Payment Gateway:** Processes payments and returns status.
4. **Sanity CMS:**
    - Stores order records with all associated metadata (e.g., timestamps, product details).

---

3. API Requirements

Detailed API Documentation

Endpoints	Method	Payload	Response	Description
/api/auth/register	POST	{ email, password }	{ message, token }	Registers a new user.
/api/auth/login	POST	{ email, password }	{token}	Logs in an existing user.
/api/products	GET	{ category, search}	[{ id, name, price, ... }]	Fetches products.
/api/products/:id	GET	-	{ id, name, description }	Retrieves product details.
/api/orders	POST	{ userId, products, total }	{ orderId, status }	Places a new order.

---

4. Sanity Schema Design

Products Schema

Description: Defines the structure for storing product details in Sanity CMS.

```
import { defineType,defineField } from "sanity";

export const product = defineType({
  name: 'product',
  type: 'document',
  title: 'Product',
  fields: [
    defineField({
      name: 'name',
      type: 'string',
      title: 'Product Name',
    }),
    defineField({
      title: 'Slug',
      name: 'slug',
      type: 'slug',
      options: {
```

```
      source: 'name',
      maxLength: 200, // will be ignored if slugify is set
      slugify: input => input
                          .toLowerCase()
                         .replace(/\s+/g, '-')
                          .slice(0, 200)
    }
  }),
  defineField({
    name: 'description',
    type: 'string',
    title: 'Description'
  }),
  defineField({
    name: 'price',
    type: 'number',
    title: 'Product Price',
  }),
  defineField({
    name: 'discountPercentage',
    type: 'number',
    title: 'Discount Percentage',
  }),
  defineField({
    name: 'isFeaturedProduct',
    type: 'boolean',
    title: 'Is Featured Product',
    description: 'Indicates if the product is featured',
    initialValue: false, // Default value
  }),
  defineField({
    name: 'stockLevel',
    type: 'number',
    title: 'Stock Level',
    description: 'Number of items in stock',
  }),
  defineField({
    name: 'category',
    type: 'string',
    title: 'Category',
    description: 'Add Category from the listed categories'
  }),
  defineField({
    name: 'image',
    type: 'image',
```

```
      title: 'Product Image',
      options: {
        hotspot: true // Enables cropping and focal point selection
      }
    })
  ]
});
```

Orders Schema

Description: Captures order details and links them to users and products.

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

---

5. Collaboration Notes

Challenges Faced

- **Tool Setup:** Initial configuration of Thunder Client for API testing and debugging.
- **Schema Design:** Balancing simplicity and extensibility for the Sanity schemas.

Feedback Incorporated

- Enhanced clarity in API responses.
- Improved usability of workflows through feedback testing.

Improvements for Future Phases

- Explore GraphQL for more efficient querying of Sanity data.
- Add collaborative features like team access for Sanity CMS.
