Marketplace Technical Foundation - Hekto

Table of Contents

---

1. System Architecture Overview

Frontend

Built with modern JavaScript frameworks (e.g., Next.js) for dynamic rendering and enhanced performance.

Responsible for fetching and displaying product data, user interfaces, and routing between pages.

Implements API calls to fetch data from the backend or directly from Sanity CMS.

Sanity CMS

Acts as the primary content management system to store and manage product information, categories, images, and metadata.

Provides APIs for querying product details, updates, and schema management.

Ensures real-time content updates that reflect immediately on the frontend.

Backend

A Node.js or serverless architecture that acts as a middle layer between the frontend, CMS, and third-party services.

Handles API requests for user authentication, order processing, and payment validation.

Manages business logic, including discounts, sales processing, and stock availability updates.

Third-Party APIs

Payment Gateway: Manages secure payment transactions, integrating with providers like Stripe or PayPal.

Authentication Service: Verifies user credentials and manages secure sessions (e.g., Firebase Authentication or Auth0).

Email Service: Sends order confirmations or promotional updates to customers.

Data Flow

Product Data: The frontend retrieves product details directly from the Sanity CMS using GROQ or GraphQL queries.

User Data: User actions (e.g., adding products to the cart) trigger API calls that interact with the backend and database.

Order Data: Order details flow from the frontend to the backend for processing, which then updates both the database and CMS.

Communication Protocols

RESTful APIs or GraphQL are utilized for seamless data exchange between components.

Webhooks notify the backend of updates in Sanity CMS (e.g., stock changes or new product entries).*(Include a clear, high-resolution diagram here illustrating the flow: User > Frontend > Sanity CMS > Third-Party APIs)*

---

2. Key Workflows

Workflow 1: User Registration

1. **Frontend:** User fills in the registration form (name, email, password, etc.).
2. **API Endpoint:** Sends the registration data to `/api/register`.
3. **Database:** User information is stored in the authentication system.
4. **Confirmation:** A verification email is sent to the user.

Workflow 2: Product Browsing

1. **Frontend:** Users navigate to categories (e.g., Home, Office, Outdoor).
2. **Sanity CMS:** Fetches product data based on the selected category.
3. **Frontend:** Displays product listings with details such as price, availability, and tags.

Workflow 3: Order Placement

1. **Frontend:** User adds products to the cart and proceeds to checkout.
2. **API Endpoint:** `/api/create-order` validates the cart details.
3. **Sanity CMS:** Updates product stock.
4. **Third-Party API:** Payment gateway processes the payment.
5. **Confirmation:** Order summary and tracking details are displayed to the user.

---

3. API Requirements

Endpoint	Method	Payload	Response
`/api/register`	POST	`{ "name": "string", "email": "string", "password": "string" }`	`{ "status": "success", "message": "User registered." }`
`/api/login`	POST	`{ "email": "string", "password": "string" }`	`{ "token": "string" }`
`/api/create-order`	POST	`{ "cart": [productIds], "userId": "string" }`	`{ "orderId": "string", "status": "confirmed" }`
`/api/products`	GET	`?category=string`	`[ { "title": "string", "price": "number" } ]`

---

4. Sanity Schema Design

Product Schema

```
export default defineType({
  name: "product",
  title: "Product",
  type: "document",
  fields: [
    { name: "title", type: "string", title: "Title" },
    { name: "price", type: "number", title: "Price" },
    { name: "category", type: "string", title: "Category" },
    { name: "stock", type: "string", title: "Stock" }
  ]
});
```

Order Schema

```
export default defineType({
  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: "totalAmount", type: "number", title: "Total Amount" },
    { name: "status", type: "string", title: "Order Status" }
  ]
});
```

---

5. Collaboration Notes

- **Peers:** Worked with the development and design teams to finalize workflows and schemas.
- **Challenges:** Ensuring seamless integration between Sanity CMS and Next.js.
- **Feedback:** Incorporated user testing feedback to refine the product browsing experience.