Marketplace Technical Foundation - SHOP.CO

Table of contents

1. System Architecture Overview

Components:

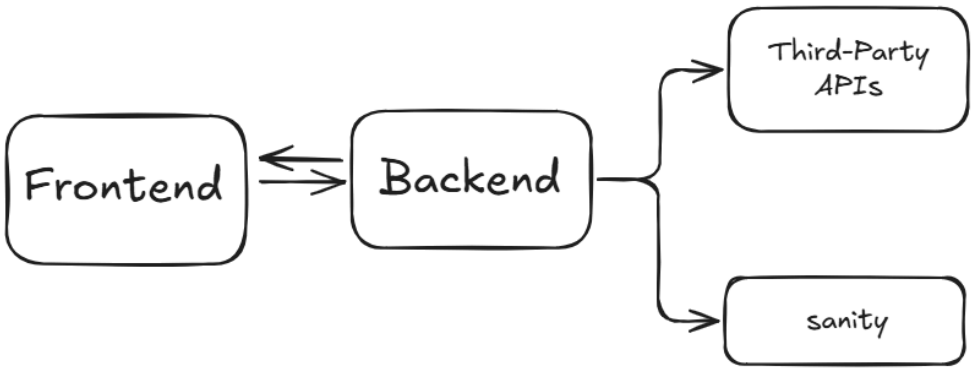
1. Frontend:
- Built using **Next.js** for a responsive and user-friendly interface.
- Handles user interactions like browsing products, adding to cart, and checkout.

2. Backend:
- Built using **Next.js** and **Sanity** for handling business logic and API requests.
- Manages user authentication, order processing, and inventory updates.

3. Sanity CMS:
- Used for managing product data, including details like name, price, description, and images.
- Provides a flexible schema for products, categories, and inventory.

4. Third-Party APIs:
- **Payment Gateway API:** Handles secure payment processing (e.g., Stripe or PayPal).
- **Authentication API:** Manages user login and registration (e.g., Firebase Auth).
- **Shipping API:** Tracks delivery status and provides shipment details.

System Architecture Diagram:

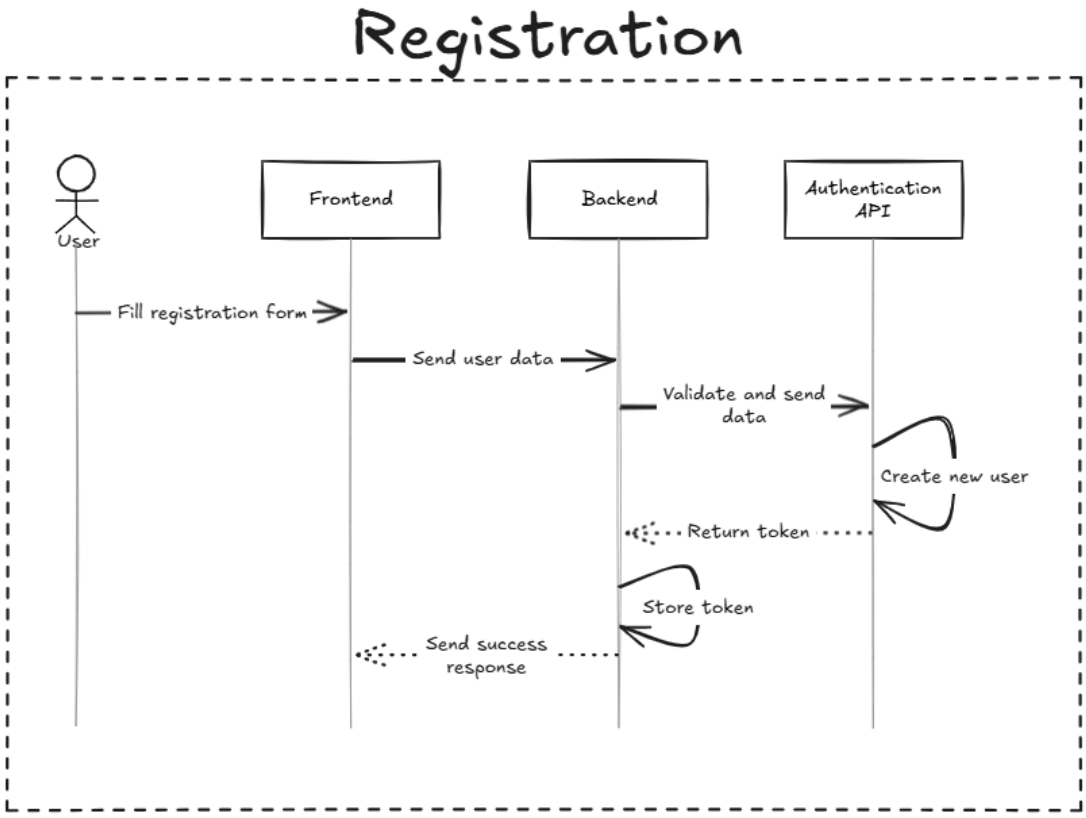


Frontend interacts with the **Backend**, which communicates with **Sanity CMS** for product data and **Third-Party APIs** for payments, authentication, and shipping.

2. Key Workflows

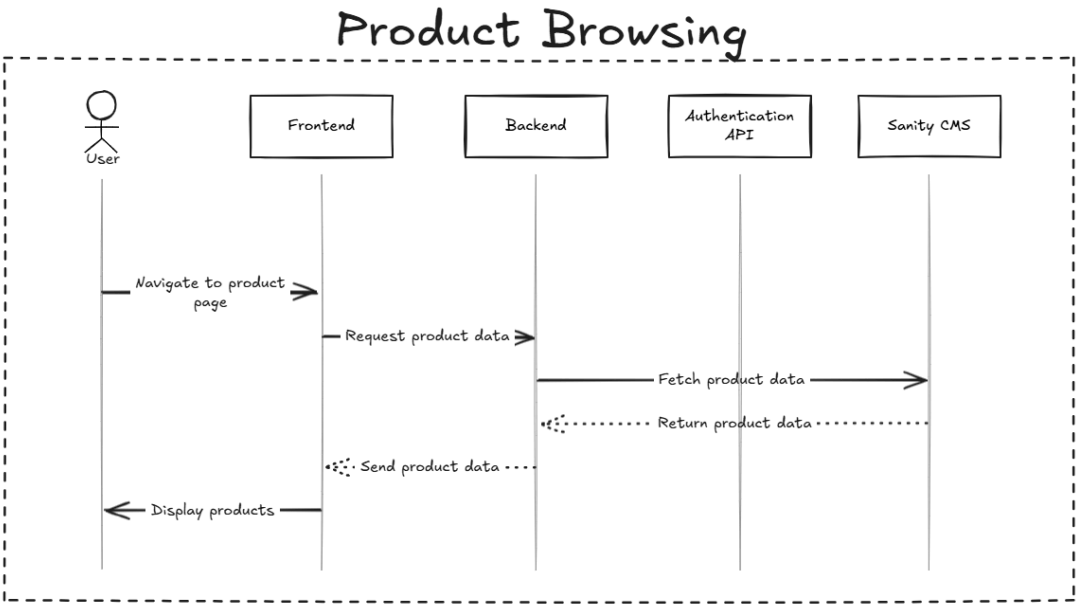
1. User Registration:

1. The user fills out the registration form on the front end.
2. Frontend sends user data to the backend.
3. Backend validates the data and sends it to the **Authentication API**.
4. Authentication API creates a new user and returns a token.
5. The backend stores the token and sends a successful response to the front end.



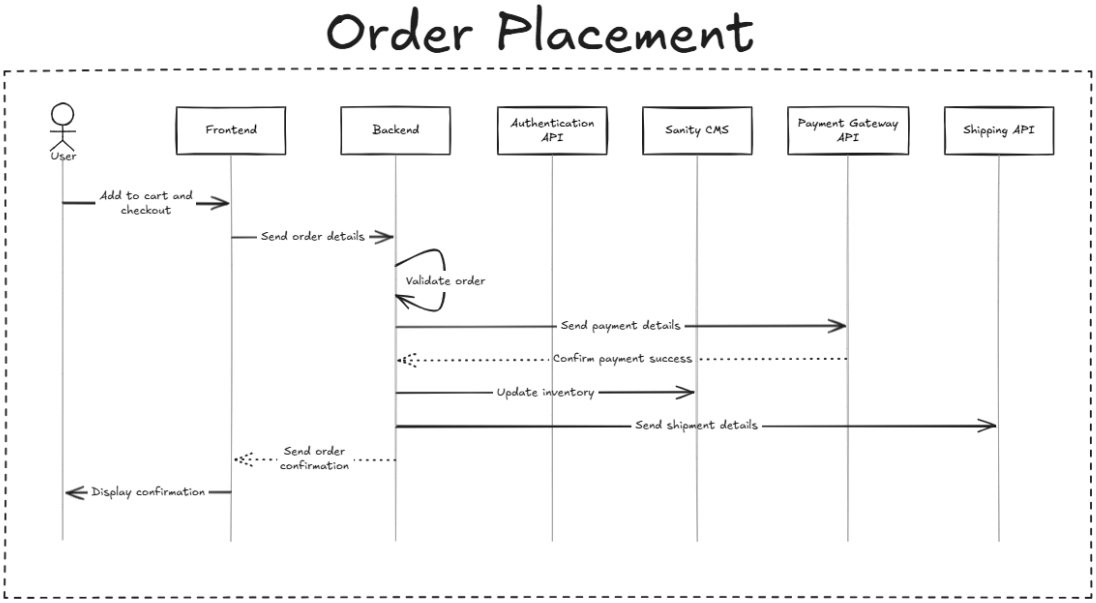
2. Product Browsing:

1. The user navigates to the product page on the front end.
2. The front end requests product data from the backend.
3. Backend fetches product data from **Sanity CMS**.
4. The backend sends the product data to the frontend for display.



3. Order Placement:

1. The user adds products to the cart and proceeds to checkout.
2. Frontend sends order details to the backend.
3. Backend validates the order and sends payment details to the **Payment Gateway API**.
4. Payment Gateway API processes the payment and confirms success.
5. Backend updates the inventory in **Sanity CMS** and sends shipment details to the **Shipping API**.
6. The backend sends an order confirmation to the frontend.



3. API Requirements

1. Authentication API:

- **Endpoint:** `/api/auth/register`
  - **Method:** POST
  - **Payload:**

```
    {
      "name": "John Doe",
      "email": "john@example.com",
      "password": "password123"
    }
    ```
  - Response:
    ```json
    {
      "token": "abc123xyz",
      "userId": "12345"
    }
```

2. Payment Gateway API:

- **Endpoint:** `/api/payment/process`
  - **Method:** POST
  - **Payload:**

```
    {
      "amount": 100,
      "currency": "USD",
      "paymentMethod": "card_12345"
    }
    ```
  - Response:
    ```json
    {
      "status": "success",
      "transactionId": "txn_67890"
    }
```

3. Shipping API:

- **Endpoint:** `/api/shipping/track`
  - **Method: GET**

- Payload:

```
    {
      "orderId": "12345"
    }
    ```
  - **Response:**
    ```json
    {
      "status": "shipped",
      "trackingNumber": "TN123456789"
    }
```

4. Sanity Schema Design

Product Schema:

```
{
  "name": "product",
  "type": "document",
  "fields": [
    { "name": "productId", "type": "string", "title": "Product ID" },
    { "name": "name", "type": "string", "title": "Name" },
    { "name": "category", "type": "string", "title": "Category" },
    { "name": "price", "type": "number", "title": "Price" },
    { "name": "description", "type": "text", "title": "Description" },
    { "name": "images", "type": "array", "of": [{ "type": "image" }],
"title": "Images" },
    { "name": "availableQuantity", "type": "number", "title": "Available
Quantity" },
    { "name": "tags", "type": "array", "of": [{ "type": "string" }],
"title": "Tags" }
  ]
}
```

Order Schema:

```
{
  "name": "order",
  "type": "document",
  "fields": [
```

```
    { "name": "orderId", "type": "string", "title": "Order ID" },
    { "name": "customerId", "type": "string", "title": "Customer ID" },
    { "name": "products", "type": "array", "of": [{ "type": "reference",
"to": [{ "type": "product" }] }], "title": "Products" },
    { "name": "totalAmount", "type": "number", "title": "Total Amount" },
    { "name": "status", "type": "string", "title": "Status" }
  ]
}
```

5. Collaboration Notes

- Team Collaboration:
- Worked closely with peers to design the system architecture and workflows.
- Divided tasks based on expertise (e.g., frontend, backend, API integration).

- Challenges Faced:
- Integrating Sanity CMS with the backend required additional research and testing.
- Ensuring secure payment processing was a priority, leading to multiple iterations.

- Feedback Incorporated:
- Improved the user registration workflow based on feedback to make it more intuitive.
- Simplified the product schema to ensure scalability.

Final Notes

- This document serves as the technical foundation for SHOP.CO.
- It ensures clarity and attention to detail, which are critical for the success of the marketplace.