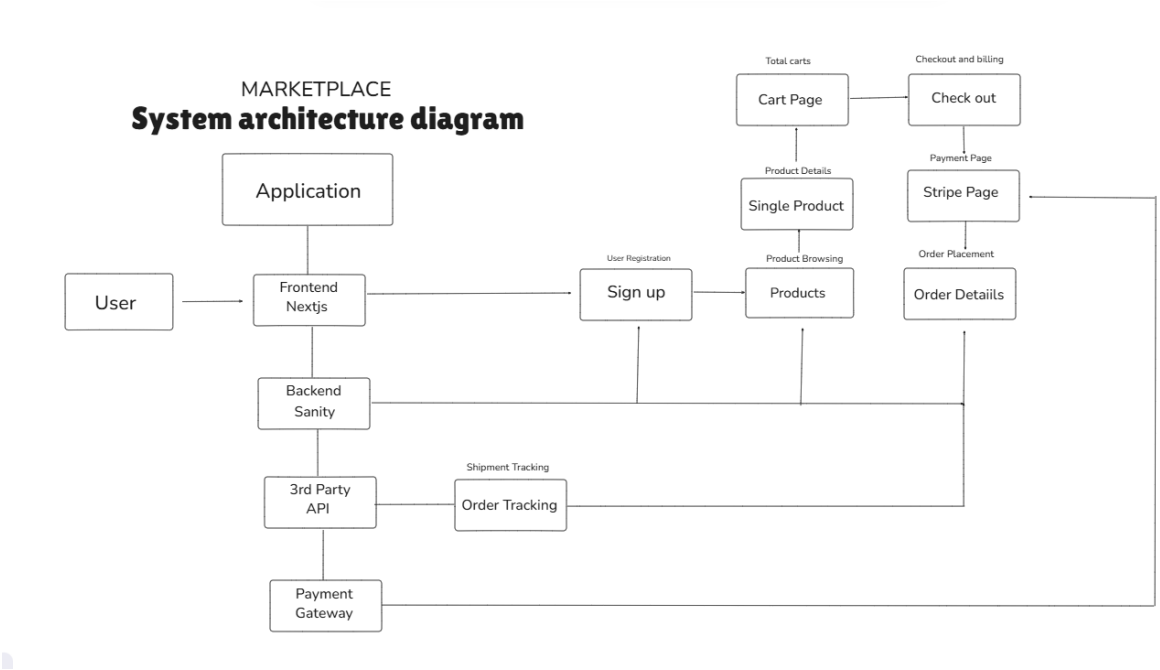
Marketplace Technical Foundation - General E-commerce

This document outlines the technical foundation required for building a general e-commerce marketplace, focusing on both frontend and backend development, and aligning with the system design architecture.

System architecture diagram



Technical Requirements

1. Frontend Development

Key Technologies:

- **Next.js:** For building the user interface with server-side rendering (SSR) to ensure faster load times and SEO benefits.
- **Tailwind CSS:** For styling the UI with a responsive and mobile-first approach.
- **TypeScript:** For static type checking and enhanced development experience.
- **Responsive Design:** To ensure compatibility across all devices (mobile, tablet, desktop).

---

2. Backend Development

Key Technologies:

- **Sanity CMS:** Serves as the backend for content management, including managing e-commerce data (products, orders, and customer information).
- **3rd Party APIs:**
    - **Payment Gateway (Stripe):** For secure payment processing.
    - **Shipment Tracking (e.g., ShipEngine):** For real-time order tracking and delivery updates.

API Endpoints

1. Fetching Products Data

Endpoint: `/products`
Method: `GET`
Description: Retrieves a list of all available products from Sanity CMS, including essential product details like stock levels, prices, and images.

Headers:

- `Authorization: Bearer <token>` (optional, for private product data access)

Query Parameters (Optional):

- `limit`: Number of products to fetch (e.g., `?limit=10`)
- `category`: Filter by product category (e.g., `?category=electronics`)
- `sort`: Sorting criteria (e.g., `?sort=price_asc`)

Response:

```
[
  {
    "id": "12345",
    "name": "Wireless Earbuds",
    "price": 49.99,
    "stock": 100,
    "image": "https://example.com/images/earbuds.jpg"
  },
  {
    "id": "67890",
    "name": "Smartphone",
    "price": 699.99,
    "stock": 25,
    "image": "https://example.com/images/smartphone.jpg"
  }
]
```

---

2. User Registration and Sign-In

Description: User registration and sign-in functionality are seamlessly handled by Clerk. It provides secure authentication and an integrated sign-up and sign-in experience without requiring additional custom endpoints.

---

3. Creating an Order

Endpoint: `/order`
Method: `POST`
Description: Creates a new order and stores it in Sanity CMS. Processes order details, including customer information, payment status, and product details.

Headers:

- `Content-Type: application/json`
- `Authorization: Bearer <token>`

Payload:

```
{
  "customerInfo": {
    "name": "Jane Doe",
    "email": "janedoe@example.com",
    "address": "456 Avenue, City, Country"
  },
  "paymentStatus": "paid",
  "productDetails": [
    {
      "productId": "12345",
      "quantity": 2,
      "price": 49.99
    },
    {
      "productId": "67890",
      "quantity": 1,
      "price": 699.99
    }
  ]
}
```

Response:

```
{
  "orderId": "order123456",
  "totalAmount": 799.97,
  "orderDate": "2025-01-16T14:25:00Z"
}
```

---

4. Order Tracking

Endpoint: `/shipment/:order_id`
Method: `POST`

Description: Tracks the shipment of an order using `order_id`. This endpoint interacts with the ShipEngine API to provide real-time shipment status and expected delivery date.

Headers:

- `Authorization: Bearer <token>`

Path Parameters:

- `order_id` (string): Unique identifier for the order to be tracked.

Payload:

```
{
  "orderId": "order123456",
  "shipmentId": "ship123456",
  "expectedDeliveryDate": "2025-01-20",
  "status": "in_transit"
}
```

Response:

```
{
  "orderId": "order123456",
  "shipmentId": "ship123456",
  "status": "in_transit",
  "expectedDeliveryDate": "2025-01-20",
  "lastUpdated": "2025-01-16T12:00:00Z"
}
```

Workflow Alignment with System Design

User Actions:

1. **Sign Up:**
   - Registers using Clerk for sign-up and sign-in functionality.
   - User data is securely handled by Clerk.
2. **Browsing Products:**
   - Fetches product data via the `/products` endpoint.
   - Supports filtering by categories and sorting.
3. **Adding to Cart:**
   - Interacts with the "Cart Page" for viewing and managing selected products.
4. **Checkout:**
   - Completes payment through the Stripe integration (via the "Stripe Page").
5. **Order Placement:**
   - Places an order using the `/order` endpoint.
   - Processes payment status and order details.
6. **Order Tracking:**
   - Tracks shipment status via the `/shipment/:order_id` endpoint, integrating with ShipEngine.

Admin and CMS Interaction:

- **Product Management:**
  - Managed via Sanity CMS to add, update, and delete products.
- **Order Management:**
  - Admin can view and process orders stored in Sanity CMS.

Created by Nadeem Khan