Marketplace Technical Foundation

This document outlines the core technical framework for the marketplace platform. It details key routes, API integrations, and the workflow for order processing.

---

tech stak

Frontend:

1. Built using Next.js, providing server-side rendering for enhanced performance and faster UI development.
2. Offers a seamless user experience with optimized page load times and improved SEO capabilities.

Backend:

1. Powered by Sanity CMS, enabling efficient management and storage of content, including products, orders, and customer data,
2. Ensures scalability and flexibility for handling various content types.

Third-Party APIs:

1. The marketplace integrates with external APIs to enhance functionalities, such as payment processing, shipping rate calculations, and product data synchronization.

Operational Workflow

Home Page

- Retrieves and displays a list of available products by fetching data from an external API.
- Users can navigate to individual product pages for detailed information.
- Includes search and filtering options to help users find specific products easily.
- Highlights featured or promotional items on the homepage for increased visibility.

Product Detail Page (`/products/{product_id}`)

- Fetches and presents detailed product information, such as:
    - **Description:** Overview of the product's features.
    - **Price:** Current pricing, including discounts or offers.
    - **Stock:** Real-time availability of the product.
- Allows users to add products to their shopping cart with a single click.
- Supports displaying user reviews and ratings to guide purchasing decisions.

Shopping Cart (`/cart`)

- Displays all items that users have added to their cart, along with:
    - Product details (e.g., name, price, and quantity).
    - The total price of all items, including taxes or shipping fees.
- Provides functionality to:
    - Add new products to the cart.
    - Update product quantities.
    - Remove unwanted items from the cart.
- Data is securely managed and stored using the Cart Schema in Sanity CMS to ensure persistence across sessions.

Checkout Page (`/checkout`)

- Guides users through the checkout process, which includes:
    - Entering or selecting shipping details.
    - Reviewing the final cart summary, including itemized costs and estimated delivery timelines.
- Integrates with payment gateways for secure payment processing.
- Displays a confirmation message once the order is successfully placed.

Order Management

- A Shipping ID is generated automatically once an order is confirmed.
- The order status is updated through the following stages:
    1. **Processing:** The order is being prepared.
    2. **Shipped:** The order has left the warehouse, and a tracking ID is issued.
    3. **Delivered:** The order has been delivered to the customer.

Order Tracking (`/order/{order_id}`)

- Users can track the status of their order using the provided tracking ID.
- Order details and current statuses are fetched from the Order Schema within Sanity CMS.

API Endpoints with Sanity Schema Exampleskment

API: Products (GET)

- **Endpoint:** `/api/products`
- **Description:** Retrieve a list of products from the Sanity CMS.
- **Sanity Schema Example:**

```
export const productSchema = {
  name: 'product',
  type: 'document',
  fields: [
    { name: 'name', type: 'string', title: 'Product Name' },
    { name: 'price', type: 'number', title: 'Price' },
```

```
    { name: 'description', type: 'text', title: 'Description' },
    { name: 'stock', type: 'number', title: 'Stock' },
    { name: 'image', type: 'image', title: 'Product Image' },
  ],
};
```

API: Orders (POST)

- **Endpoint:** `/api/orders`
- **Description:** Create a new order in the system.
- **Sanity Schema Example:**

```
export const orderSchema = {
  name: 'order',
  type: 'document',
  fields: [
    { name: 'orderId', type: 'string', title: 'Order ID' },
    { name: 'products', type: 'array', of: [{ type: 'reference', to:
{ type: 'product' } }], title: 'Products' },
    { name: 'total', type: 'number', title: 'Total Price' },
    { name: 'status', type: 'string', title: 'Order Status', options:
{ list: ['Processing', 'Shipped', 'Delivered'] } },
    { name: 'customer', type: 'string', title: 'Customer Name' },
  ],
};
```

API: Shipment (GET)

- **Endpoint:** `/api/shipment`
- **Description:** Track the shipment status of an order using a third-party API.
- **Sanity Schema Example:**

```
export const shipmentSchema = {
  name: 'shipment',
  type: 'document',
  fields: [
    { name: 'trackingId', type: 'string', title: 'Tracking ID' },
    { name: 'orderId', type: 'reference', to: { type: 'order' },
title: 'Order ID' },
    { name: 'status', type: 'string', title: 'Shipment Status',
options: { list: ['In Transit', 'Out for Delivery', 'Delivered'] } },
    { name: 'estimatedDelivery', type: 'datetime', title: 'Estimated
Delivery' },
  ],
};
```

Diagram

Below is a high-level flow diagram illustrating the route workflow:

- **Home Page →Product Detail Page →Shopping Cart →Checkout →Order Tracking**

Additional Considerations

1.

Authentication and User Accounts:

2.

- Users can create accounts to save their order history and manage preferences.
- Implement secure login and registration processes using industry-standard authentication protocols.

3.

Performance Optimization:

4.

- Implement caching mechanisms and content delivery networks (CDNs) for faster page loading.
- Optimize API calls to reduce latency.

5.

Mobile Responsiveness:

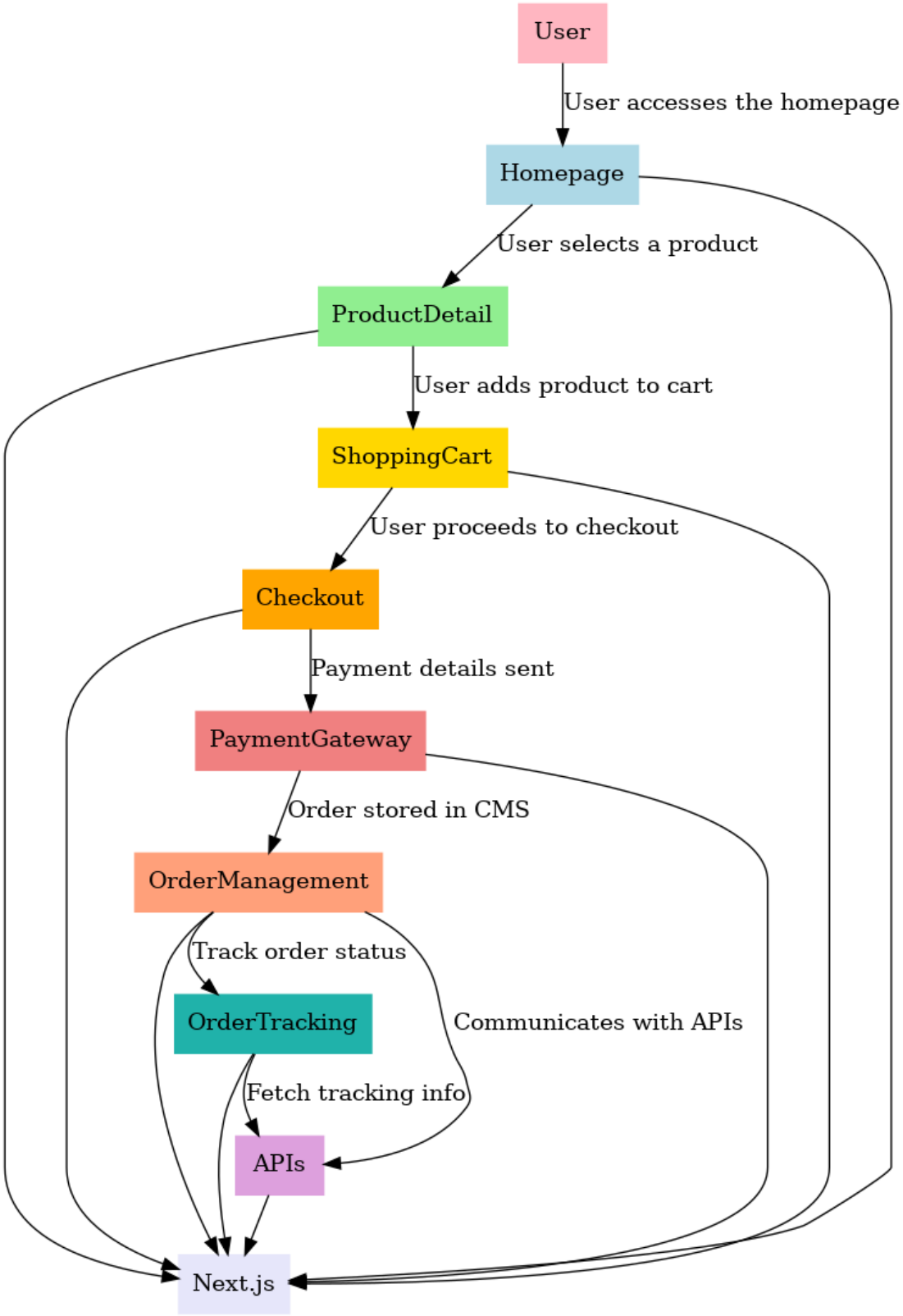
6.

- Ensure all pages and components are fully responsive for seamless mobile and tablet experiences.

Analytics and Monitoring:

7.

- Integrate tools like Google Analytics to track user behavior and platform performance.
- Use monitoring services to detect and resolve issues in real-time.


User
User accesses the homepage
Homepage
User selects a product
ProductDetail
User adds product to cart
ShoppingCart
User proceeds to checkout
Checkout
Payment details sent
PaymentGateway
Order stored in CMS
OrderManagement
Track order status
OrderTracking
Communicates with APIs
Fetch tracking info
APIs
Next.js