Technical Foundation: Marketplace Project Day 2

1. Overview

The primary objective of this technical foundation is to transition from the business-oriented planning of Day 1 to creating a robust technical framework for the marketplace. This involves defining the system architecture, planning API requirements, and documenting workflows and data schemas. The project uses Sanity CMS for backend management and integrates third-party APIs for payment and shipment tracking.

2. Technical Requirements

Frontend:

- **Framework:** Next.js
  - A React-based framework for building server-rendered and statically generated web applications.
- **Styling:** Tailwind CSS
  - A utility-first CSS framework for rapid UI development.
- **Component Library:** shadcn
  - A collection of accessible and customizable components for consistent UI design.
- **Shipping Integration:** ShipEngine
  - An API for shipping label creation, rate calculation, and tracking packages.
- **Payment Integration:** Stripe
  - A secure payment gateway for handling online transactions.

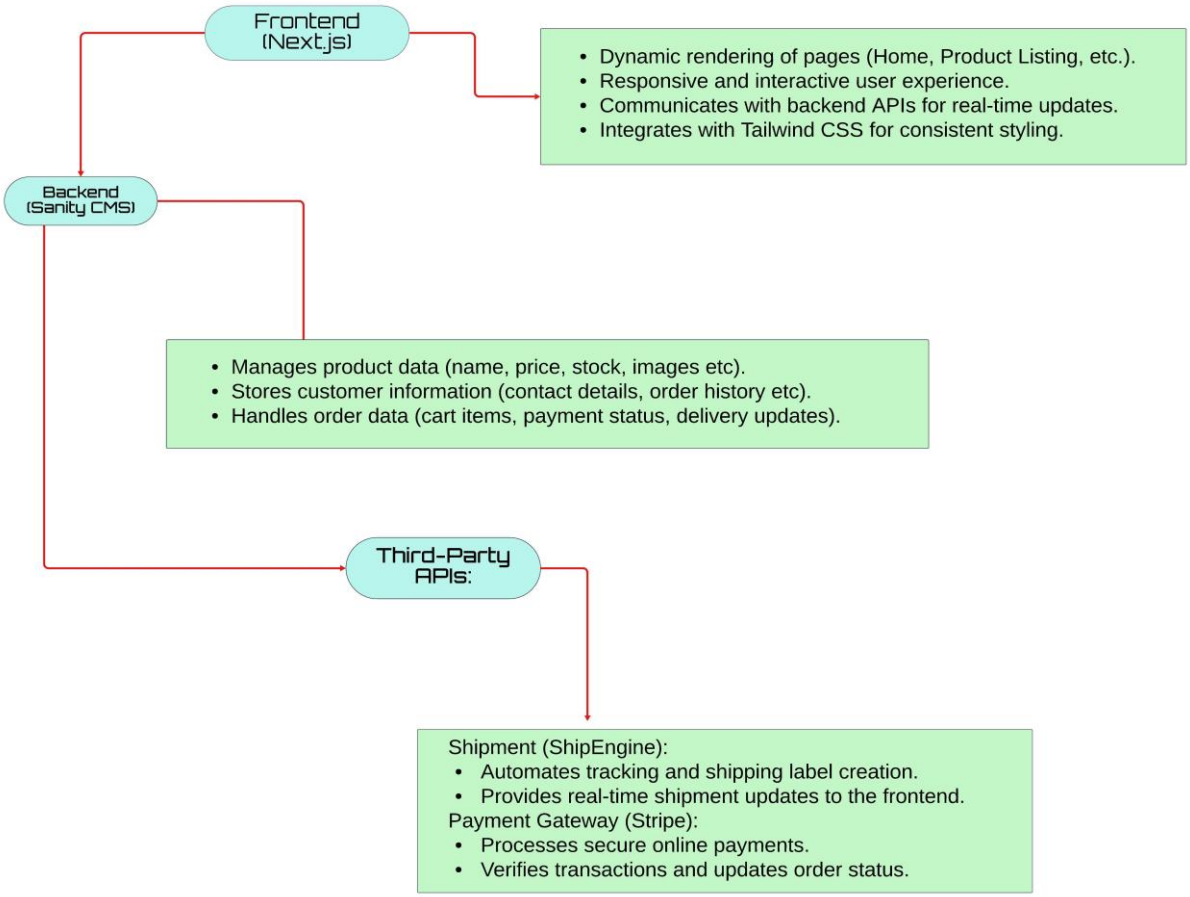
Pages:

- **Home:**
  - The landing page showcasing the marketplace's purpose, featured products, and key highlights.
- **Product Listing:**
  - Displays a list of all available products with filtering and sorting options.
- **Product Details:**
  - Provides detailed information about a specific product, including images, price, and description.
- **Cart:**
  - Allows users to view and manage selected products before proceeding to checkout.
- **Checkout:**
  - A page for users to enter shipping details and complete the payment process.
- **Order Confirmation:**
  - Displays order details and confirmation after a successful purchase.

Backend:

- **CMS:** Sanity CMS
  - Provides a flexible and efficient way to manage content such as products, orders, and customers.
  - Enables developers to design and customize schemas to meet specific business needs.
  - Supports real-time updates, ensuring the frontend reflects the latest data instantly.
- **Third-Party APIs:**
  - **ShipEngine:**
    - Integrates shipping functionalities, including creating labels and tracking packages.
    - Reduces complexity by automating shipping rate calculations.
    - Provides real-time shipment updates to users.
  - **Stripe:**
    - Manages secure payment processing for various payment methods.
    - Handles complex financial transactions with fraud protection.
    - Provides detailed payment status updates for seamless order management.

3. System Architecture Design



4. API Requirements

General API Endpoints

1. **Products**
   - **Endpoint:** `/products`
   - **Method:** GET
   - **Description:** Fetch all available products from Sanity.
   - **Response Example:** `{ "id": 1, "name": "Product A", "price": 100 }`
2. **Orders**
   - **Endpoint:** `/orders`
   - **Method:** POST
   - **Description:** Create a new order in Sanity.
   - **Payload Example:** `{ "customer": "John Doe", "items": [...], "paymentStatus": "Paid" }`
3. **Shipment**
   - **Endpoint:** `/shipment`
   - **Method:** GET
   - **Description:** Fetch order tracking status.
   - **Response Example:** `{ "orderId": 123, "status": "In Transit", "ETA": "2 days" }`

5. Sanity Schemas

Example: Product Schema

```
export default {
  name: 'product',
  type: 'document',
  fields: [
    { name: 'name', type: 'string', title: 'Product Name' },
    { name: 'price', type: 'number', title: 'Price' },
    { name: 'stock', type: 'number', title: 'Stock Level' },
    { name: 'description', type: 'text', title: 'Description' },
    { name: 'image', type: 'image', title: 'Product Image' }
  ]
};
```