
Day 2: Planning the Technical Foundation

Day 2 Goal

The goal of Day 2 is to transition from business planning to technical preparation, establishing the foundation required to build an electronics e-commerce marketplace. This document outlines the technical requirements, system architecture, API endpoints, and Sanity CMS schema essential for building a functional and scalable platform.

Recap of Day 1: Business Focus

Business Goals Defined:

- **Problem Solved:** Simplifying access to a wide range of electronics products for customers by offering a user-friendly platform.
- **Target Audience:** Tech-savvy individuals, professionals, and electronics enthusiasts.
- **Products Offered:** Smartphones, laptops, accessories, home appliances, and gadgets.
- **Unique Value Proposition:** Competitive pricing, seamless user experience, and comprehensive product information.

Data Schema Drafted:

- Core entities: Products, Orders, Customers, Delivery Zones, Shipments.

Define Technical Requirements

Frontend Requirements:

- Framework: **Next.js** for a responsive, dynamic, and SEO-friendly interface.
- Key Pages:
    - **Homepage**: Showcase featured electronics and categories.
    - **Product Listing**: Allow filtering by category, brand, and price.
    - **Product Details**: Display specifications, reviews, and availability.
    - **Cart**: Enable item review and adjustments.
    - **Checkout**: Secure order placement with payment.
    - **Order Confirmation**: Provide details of successful orders.

Backend Requirements (Sanity CMS):

- Content management for:
    - Product data (names, prices, specifications, images).
    - Customer details and order records.
    - Categories and tags for easy filtering.
- Real-time synchronization with frontend for dynamic updates.

Third-Party API Integrations:

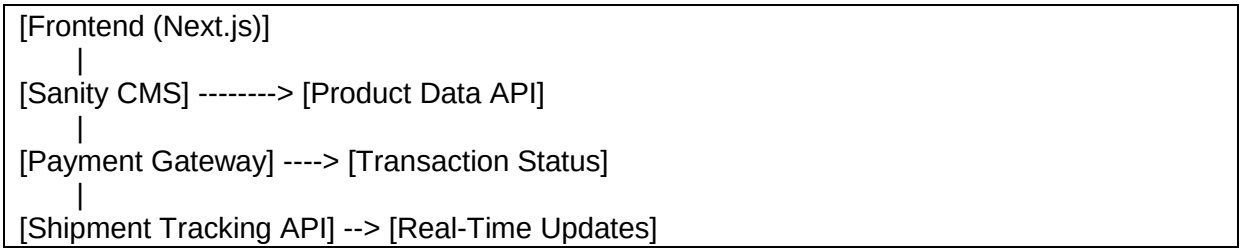
- **Payment Gateway:** Secure payment processing (e.g., Stripe or PayPal).
- **Shipment Tracking API:** Real-time tracking of orders.
- **Email Service API:** Notifications for order confirmation and updates.

System Architecture

Overview

1. **Frontend (Next.js):** User interface for browsing, searching, and purchasing electronics.
2. **Sanity CMS:** Backend for managing product information, orders, and customer data.
3. **Third-Party APIs:** Integration for payment processing and shipment tracking.

System Workflow Diagram



Key Workflows:

1. **User Registration:**
    - User signs up -> Data stored in Sanity -> Confirmation sent via email.
2. **Product Browsing:**
    - User views products -> Sanity API fetches data -> Displayed dynamically.
3. **Order Placement:**
    - User adds items to cart -> Proceeds to checkout -> Order saved in Sanity.
4. **Shipment Tracking:**
    - Status updates fetched via shipment API -> Displayed on user dashboard.

Plan API Requirements

General API Endpoints

Endpoint	Method	Description	Payload/Response
/products	GET	Fetch all available products	Response: { id, name, price, stock, image }
/product/{id}	GET	Fetch details of a single product	Response: { id, name, specs, reviews }
/orders	POST	Create a new order	Payload: { customerInfo, items, payment }
/orders/{id}	GET	Fetch details of an order	Response: { id, status, items, shipment }
/shipment/{id}	GET	Track shipment status via API	Response: { id, status, eta }

API Example

Endpoint: `/products`

- **Method:** GET
- **Description:** Fetch all products from Sanity CMS.
- **Response:**

```
[
  {
    "id": 1,
    "name": "Smartphone XYZ",
    "price": 699,
    "stock": 120,
    "image": "smartphone_xyz.jpg"
  }
]
```

Sanity Schema Draft

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: 'image', type: 'image', title: 'Product Image' },
    { name: 'category', type: 'string', title: 'Category' },
    { name: 'tags', type: 'array', of: [{ type: 'string' }], title: 'Tags' },
    { name: 'description', type: 'text', title: 'Description' },
  ],
};
```

Order Schema

```
export default {
  name: 'order',
  type: 'document',
  fields: [
    { name: 'customer', type: 'reference', to: [{ type: 'customer' }], title: 'Customer' },
    { name: 'items', type: 'array', of: [{ type: 'reference', to: [{ type: 'product' }] }], title:
'Items' },
    { name: 'total', type: 'number', title: 'Total Amount' },
    { name: 'status', type: 'string', title: 'Order Status' },
    { name: 'createdAt', type: 'datetime', title: 'Order Date' },
  ],
};
```

Technical Documentation

1. **System Architecture Overview:**
    - Diagram showing interactions between frontend, CMS, and APIs.
    - Description of components and their roles.
2. **Key Workflows:**
    - Registration, browsing, order placement, and tracking workflows.
3. **API Specification:**
    - Endpoints, methods, payloads, and sample responses.
4. **Sanity Schemas:**
    - Product, Order, and Customer schemas.

Key Outcome

By the end of Day 2, we have:

1. A technical plan aligned with business goals.
2. A system architecture visualizing component interaction.
3. Detailed API requirements tailored to the marketplace.
4. Sanity schemas to handle core entities and workflows.

This technical foundation sets the stage for the implementation phase on Day 3.