**DAY 2 PLANNING THE TECHNICAL FOUNDATION**

**1. Define Technical Requirements**

The first step in building my marketplace is to translate my business goals into actionable technical requirements. This involves detailing the frontend and backend needs to ensure my platform meets user expectations and functions efficiently.

**Frontend Requirements**

1. **User-Friendly Interface**:
   * Design an intuitive and visually appealing interface to allow users to browse, search, and filter products effortlessly.
2. **Responsive Design**:
   * Ensure the platform works seamlessly across all devices (mobile, tablet, and desktop).
3. **Essential Pages**:
   * **Home Page**: Highlights featured products, categories, and promotional offers.
   * **Product Listing Page**: Displays all products with search, filters, and sorting options.
   * **Product Details Page**: Provides comprehensive information about each product (name, price, description, stock availability, images).
   * **Cart Page**: Shows selected products, quantities, and the total cost with an option to update or remove items.
   * **Checkout Page**: Facilitates secure payment and collects delivery information.
   * **Order Confirmation Page**: Displays order details and confirms successful transactions.

**Backend Requirements with Sanity CMS**

1. **Sanity CMS Integration**:
   * Use Sanity CMS as the backend to manage dynamic data for the marketplace, such as product catalogs, customer details, and order records.
2. **Schema Design**:
   * Design schemas in Sanity to match the entities defined earlier:
     + **Products**: Store details like name, price, stock, category, tags, and images.
     + **Customers**: Capture customer profiles, contact details, and order history.
     + **Orders**: Record customer orders, products purchased, and statuses.
     + **Shipments**: Track delivery zones, shipment statuses, and delivery dates.
3. **Real-Time Updates**:
   * Leverage Sanity's real-time capabilities to reflect changes instantly across the platform (e.g., stock updates or product availability).
4. **APIs for Communication**:
   * Use Sanity's API to fetch data dynamically for the frontend (e.g., product listings, order details).

**Additional Technical Features**

* **Authentication**: Secure login and registration for customers.
* **Payment Gateway Integration**: Connect with payment providers like Stripe or PayPal for secure transactions.
* **Error Handling**: Provide user-friendly error messages for any failed actions (e.g., out-of-stock products, payment issues).
* **Search Optimization**: Implement search and filtering functionality for better product discovery.
* **Authentication Pages**: Add **Login**, **Sign-up**, and **Profile Management** pages for user authentication.

**2. Design System Architecture**

[Frontend (Next.js)]

    |

    | <-- Fetch Product Data

    | --> Send Customer Data

[Sanity CMS]

    |

    | --> Provide Data to Frontend

[Product Data API]  >>>>>>>>>>>>>>>>>(not sure)

    |

    | --> Sync Product Changes

[Third-Party API]

    |

    | --> Shipment Tracking

[Payment Gateway]

    |

    | --> Process Transactions

**Marketplace Technical Foundation - [Quollex]**

**1. System Architecture Overview**

**Diagram:**

**Frontend (Next.js)** → **Sanity CMS (Manual Product Management)** → **Payment Gateway API  
Frontend (Next.js)** → **API  
Frontend (Next.js)** → **Clerk Authentication API**

**Overview:**

The system consists of three major components:

1. **Frontend (Next.js)**:
   * The user-facing interface built with Next.js, providing product browsing, cart management, and order placement functionalities.
   * It interacts with **Sanity CMS** for product data, **Clerk** for authentication, **Payment Gateway API** for transaction processing, and **Shipment Tracking API** for tracking orders.
2. **Sanity CMS**:
   * Acts as the content management system for product data, customer details, and order records.
   * Admins manually add or update products via **Sanity Studio**, and the data is served to the frontend through a read-only API.
3. **Third-Party APIs**:
   * **Payment Gateway API** (e.g., Stripe, PayPal): Manages payments and processes transactions.
   * **Shipment Tracking API**: Provides real-time shipment tracking information.
4. **Clerk Authentication API**:
   * Handles user authentication, including login, signup, and session management.

**2. Key Workflows**

**User Adds Products to Cart**

* **Step 1**:
  + The user browses the product listings (via the frontend, powered by Next.js). Products are fetched from **Sanity CMS** through a read-only API.
* **Step 2**:
  + The user selects a product to add to the cart. Product data is displayed using the details retrieved from Sanity.
* **Step 3**:
  + The cart is updated dynamically (state management on the frontend).
* **Step 4**:
  + The user proceeds to checkout. If not already authenticated, they are prompted to log in via **Clerk Authentication API**.
* **Step 5**:
  + Once authenticated, the user proceeds with payment via the **Payment Gateway API**.
* **Step 6**:
  + After payment completion, an order record is created in **Sanity CMS**, and shipment details are sent to the **Shipment Tracking API** for tracking updates.

**User Logs In (Authentication Workflow)**

* **Step 1**:
  + User clicks on the login button on the frontend. This triggers the **Clerk Authentication API** to display login options (e.g., email/password, social logins).
* **Step 2**:
  + User enters credentials or uses social login. Clerk verifies credentials and creates a session.
* **Step 3**:
  + On successful login, Clerk returns an authentication token that is stored on the client side (e.g., in cookies or local storage).
* **Step 4**:
  + The user is granted access to the cart, order history, and profile.

**3. Category-Specific Instructions**

**General eCommerce**

This marketplace falls under **General eCommerce**, where the primary workflow includes browsing products, adding items to the cart, and placing orders.

**4. Data Schema Design**

**Entities and Relationships:**

1. **Product**
   * **ID** (String)
   * **Name** (String)
   * **Price** (Number)
   * **Stock** (Number)
   * **Category** (String)
   * **Tags** (Array of Strings)
   * **Images** (Array of Strings)
   * **Description** (String)
2. **Customer**
   * **ID** (String)
   * **Name** (String)
   * **Email** (String)
   * **Address** (String)
   * **Order History** (Array of Order IDs)
3. **Order**
   * **ID** (String)
   * **Customer ID** (String)
   * **Products** (Array of Product IDs with quantities)
   * **Total Amount** (Number)
   * **Status** (String)
   * **Payment Status** (String)
   * **Shipment ID** (String)
4. **Shipment**
   * **ID** (String)
   * **Order ID** (String)
   * **Status** (String)
   * **Tracking Number** (String)
   * **Estimated Delivery** (Date)

**5. Technical Roadmap**

**Milestone 1: Initial Setup (Week 1-2)**

* Set up Next.js frontend with pages for Home, Product Listing, Product Details, Cart, Checkout.
* Integrate **Clerk Authentication API** for user login and signup.

**Milestone 2: Sanity CMS Integration (Week 2-3)**

* Define and implement Sanity schemas for Products, Customers, Orders, and Shipments.
* Manually populate product data using **Sanity Studio**.

**Milestone 3: Payment Gateway Integration (Week 3-4)**

* Integrate **Stripe API** for payment processing during checkout.
* Test successful and failed transactions.

**Milestone 4: Shipment Tracking (Week 3-4)**

* Implement **Shipment Tracking API** to provide real-time updates.
* Integrate tracking updates on the Order Confirmation page.

**Milestone 5: Final Testing & Deployment (Week 4)**

* Conduct full user acceptance testing (UAT).
* Deploy to production environment.