**Hackathon Day\_2 Market Place Technical Foundation:**

**Overview:**

This document describes the technical setup for the marketplace projects. It includes routes, API endpoints, and how to process the order. It follows best practices to ensure scalability and efficiency.

**Step 1: Define the Technical Requirements**

**Technology Stack**

1. **Frontend:** We use Next.js, a React framework that supports server-side rendering (SSR), Static site generation (SSG), and incremental site regeneration. It makes it compatible with Sanity Content Management System and it can help to delivering dynamic content efficiently.
2. **Backend:** We can manage using Sanity CMS, which provides a schema driven approach to content management and support REST APIs.
3. **Third Party APIs:** We provide external services for additional functionalities, such as product listing and details.

**Product Listing:** Endpoint (/products) providing a list of all available products.

**Product Details:** Endpoint (/products/ [product id]) providing product detailed information about specific product.

**Step 2: Design System Architecture**

**Sanity API Endpoints**

1. **Customer Schema (/customer):**
   1. **Create (POST):** Adds a new customer.
   2. **Get (GET):** Retrieves customer details.
   3. **Update (PUT):** Modifies existing customer information.
   4. **Delete (DELETE):** Removes a customer.
2. **Order Schema (/order):**
   1. **Create (POST):** Adds a new order.
   2. **Get (GET):** Retrieves order details.
   3. **Update (PUT):** Modifies an existing order.
   4. **Delete (DELETE):** Removes an order.
3. **Cart Schema (/cart):**
   1. **Create (POST):** Add items to the cart.
   2. **Get (GET):** Retrieves cart contents.
   3. **Update (PUT):** Modifies items in the cart.
   4. **Delete (DELETE):** Clears the cart.

## **Step 3: Plan API Requirements**

### **Workflow**

#### **Homepage (/)**

* **Function:** Displays a list of products fetched from a third-party API.
* **User Interaction:** Users can click on products to view more details.

#### Product Page (/products/[product\_id])

* **Function:** Shows detailed information about a selected product.
* **User Interaction:** Users can add the product to their shopping cart.

#### Cart Page (/cart)

* **Function:** Displays items the user has added to their cart.
* **User Interaction:** Users can modify the cart by adding, editing, or removing items.
* **Data Management:** Cart information is stored in Sanity CMS.

#### Checkout Page (/checkout)

* **Function:** Allows users to enter their details and review their order before finalizing the purchase.
* **Backend Actions:**
  + **Create a New Customer Record:** Saves the customer's information in Sanity.
  + **Create a New Order Record:** Saves the order details in Sanity.
  + **Assign Shipping ID:** Generates a unique identifier for shipping once the checkout is successful.

## **Step 4: Write Technical Documentation**

### **Order Processing**

1. **Processing:** The order is received and is being prepared.
2. **Shipped:** The order is dispatched, and a tracking ID is assigned.
3. **Delivered:** The order is successfully delivered to the customer.

#### Order Tracking (/order/{order\_id})

* **Function:** Allows users to track their orders using a tracking ID.
* **Data Management:** Fetches order details and current status from the Sanity CMS.

## **Step 5: Collaborate and Refine**

### **Data Schemas**

#### **Product Schema (Sanity)**

* **Product\_id:** Unique identifier for each product.
* **Name:** Name of the product.
* **Image:** URL of the product's image.
* **Price:** price of the product.
* **Description:** Description of the product.
* **Stock:** Number of items available in stock.
* **Category:** Category to which the product belongs.
* **Variants:** Options like sizes and colors.

#### Customer Schema (Sanity)

* **Customer\_id:** Unique identifier for each customer.
* **Name:** Customer's name.
* **Email:** Customer's email address.
* **Address** Customer's shipping address.
* **Phone:** Customer's contact number.

#### Order Schema (Sanity)

* **Order\_id:** Unique identifier for each order.
* **Customer\_id:** ID of the customer who placed the order.
* **Items** List of products included in the order.
* **Total\_price:** Total cost of the order.
* **Status:** Current status of the order (Processing, Shipped, Delivered).
* **Shipping\_Id:** Unique identifier for the shipping process.
* **Tracking\_id:** Tracking number for the order.

## **Step 6: Finalize and Implement**

### **Frontend Architecture**

* **Purpose:** Utilizes Next.js for its SSR, SSG, and ISR capabilities to enhance performance and SEO.
* **Routing:** Dynamic routing handles different product categories, product pages, and other parts of the website like checkout and order history.

### Content Management (Sanity CMS)

* **Schema Design:** Ensures well-structured content models for products, categories, and variants.
  + **Product Schema:** Includes fields for product name, description, price, images, category, size, color, and availability.
  + **Category Schema:** Essential for filtering products efficiently.

### API Integration

* **Efficient Data Fetching:** Sanity’s API is used to fetch product and content data, with caching implemented to improve load times.
* **Context API:** Manages product prices and other state data for a smooth user experience.

### Website Features

* **Product Listings and Filtering:** Uses a responsive grid for displaying products. Filters allow sorting by size, color, price, and other criteria.
* **Product Pages:** Displays product details, images (served responsively), and offers an add-to-cart feature.
* **Shopping Cart:** Utilizes Context API for state management. The cart page allows quantity adjustments and item removals.
* **Checkout:** Integrates with payment providers like Stripe or PayPal for secure transactions.

### Additional Functionality

* **Search Engine:** Implements robust search capabilities with filtering options.
* **User Authentication:** Supports login, registration, social logins, and manages order history and saved carts.
* **Performance Optimizations:** Includes lazy loading for images and large components, enhancing performance.

## 

## Diagram:

Below is a simple flow diagram showing how users move through the app:

+-----------+ +----------------+ +-------------+ +-----------+

| Home | → | Product Page | → | Cart Page | → | Checkout |

+-----------+ +----------------+ +-------------+ +-----------+

↓ ↓

+-----------------+ +----------------+

| Order Created | | Order Tracking |

+-----------------+ +----------------