Tab 2

Marketplace Technical Foundation - Niche E-Commerce

**16 JANUARY 2025**

# 

# Overview

The marketplace has been designed for a single business owner who is both the manufacturer and the seller. The focus is on products, customers, orders, shipping, and reviews, that leverages the product improvements and personalization

# Technical Requirements:

**User Interface (UI) Section**

**Website Design & Navigation Layout and Components:**

Built using Next.js to handle routing and server-side rendering.

Utilizing Tailwind CSS for styling, ensuring a responsive and modern design.

Includes navigation links to the home, quiz, and product sections.

**Skin Type Quiz Page**

**Description:**

"Interactive quiz for users to determine their skin type, redirecting to tailored product recommendations based on inputs".

**Product List from Quiz**

**Description:**

"Dynamically generated list of products based on quiz results, alongside additional product categories."

**Product Detail Page**

**Description:**

"Detailed page for each product containing descriptions, images, prices, and user reviews."

**Cart & Checkout Page**

**Description:**

"Allows users to review items, enter shipping details, and proceed with secure payment, including dynamic delivery charges."

**Shipping API Integration**

**Description:**

"Integrates with external shipping service APIs to calculate shipping costs dynamically based on delivery address and selected shipping options."

**Backend Services Components:**

**Database Management (Sanity CMS):**

**"Sanity CMS":**

Manages all content and product data dynamically.

**Description:**

Manages all content and product data dynamically.

Allows easy updates and querying of product information, user profiles, and orders.

Supports structured content types for products, enabling tailored displays on the frontend.

**Contains specific tables for efficient data organization:**

**Order Details:**

Stores information regarding each customer's order, including products purchased, quantities, and order status.

**Customer Details:**

Keeps records of customer profiles, including name, contact information, and order history.

**Shipping Details:**

Manages shipping information, including addresses, shipping methods, and tracking numbers. Maintains a record of fixed delivery charges based on predefined parameters.

**Skin Type Quiz:**

Holds the results of skin type quizzes taken by users to provide tailored product recommendations.

**Products:**

Contains product information like descriptions, images, pricing, and inventory levels.

**Payment Processing:**

**Description:**

Handles transactions predominantly through Cash on Delivery.

Ensures secure order placement and provides receipt generation for customers during checkout.

**Order Management:**

**Description:**

Tracks current order statuses, facilitates updates from the user interface, and manages communication with the Sanity database to reflect real-time changes in product availability and order history.

**Shipping API Integration:**

**Description:**

"Utilizes APIs to track and update shipping statuses. Integrates with order details to provide customers with real-time tracking capabilities from shipment creation to delivery."

**Example Processes Associated with Sanity CMS**

**Content Updates:**

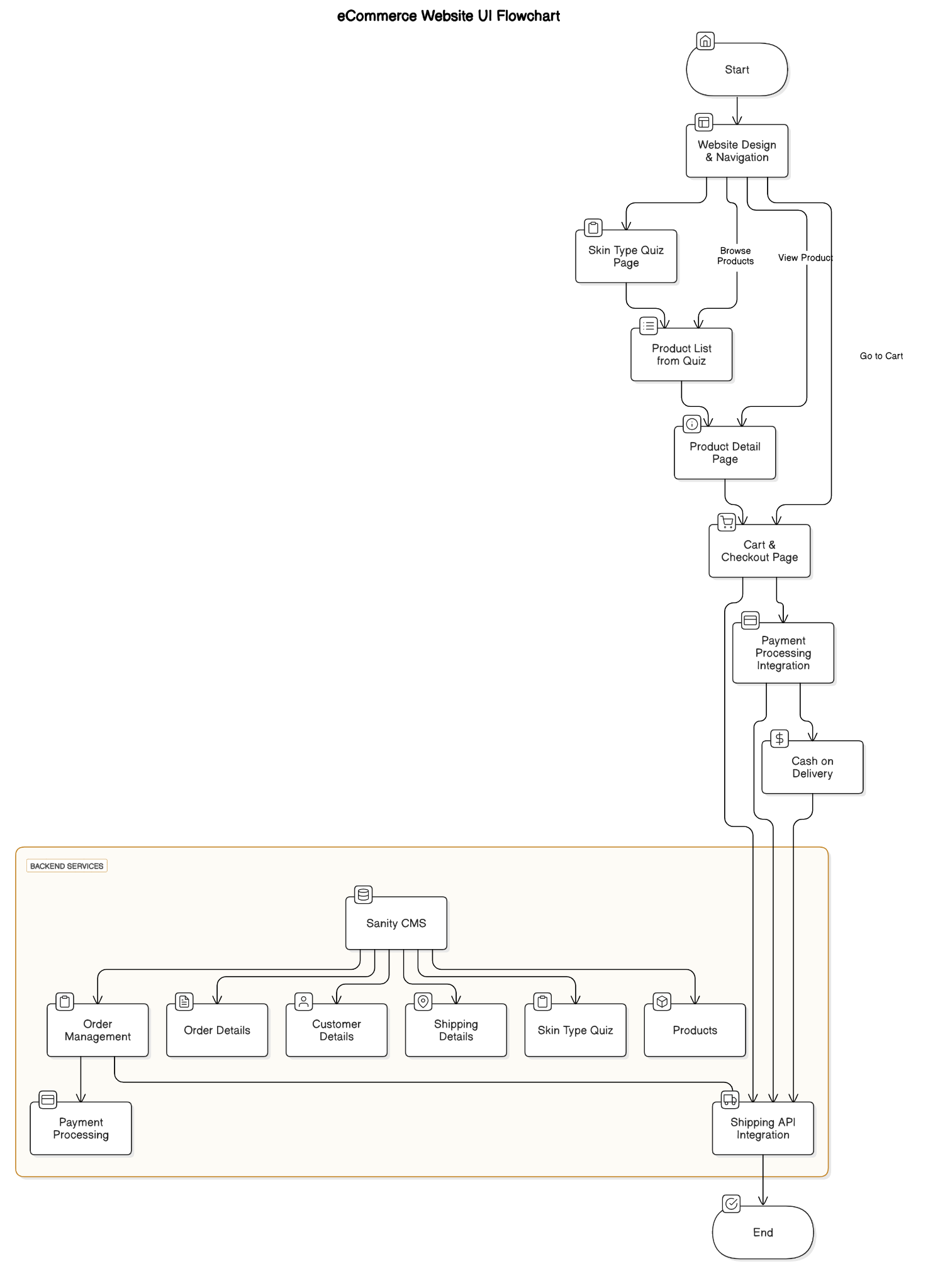
When a new product is added or information needs to be updated, staff use Sanity's interface to modify product details, which then reflect on the product listing pages dynamically through the frontend.

**Data Retrieval:**

The frontend queries the Sanity CMS to fetch data like product listing, details, and user profile info as users navigate the site. This ensures users always see the most current information.

**Order Processing Workflow:**

Upon checkout, order data is stored in the Sanity database, and order status is updated as the fulfillment progresses. Customers can view their order history and statuses in their profiles.

Data Flowchart:  


### 

### **API Endpoints:**

### **Orders**

* **Create Order**
  + Endpoint: POST /orders
  + Request Body: Order object
  + Response: Created Order object with 201 Created
* **Get Order by ID**
  + Endpoint: GET /orders/{id}
  + Response: Order object with 200 OK
* **Update Order**
  + Endpoint: PUT /orders/{id}
  + Request Body: Order object
  + Response: Order object with 200 OK
* **Delete Order**
  + Endpoint:DELETE /orders/{id}
  + Response: 204 No Content

**Payments**

* **Create Payment**
  + Endpoint:POST /payments
  + Request Body: Payment object
  + Response: Created Payment object with 201 Created
* **Get Payment by Order ID**
  + Endpoint:GET /payments/order/{orderId}
  + Response:Payment object with 200 OK
* **Update Payment**
  + Endpoint:PUT /payments/{id}
  + Request Body: Payment object
  + Response:Updated Payment object with 200 OK
* **Delete Payment**
  + Endpoint:DELETE /payments/{id}
  + Response: 204 No Content

**Shipping**

* **Create Shipping**
  + Endpoint:POST /shipping
  + Request Body: Shipping object
  + Response: Created Shipping object with 201 Created
* **Get Shipping by Order ID**
  + Endpoint:GET /shipping/order/{orderId}
  + Response: Shipping object with 200 OK
* **Update Shipping**
  + Endpoint:PUT /shipping/{id}
  + Request Body: Shipping object
  + Response:Updated Shipping object with 200 OK
* **Delete Shipping**
  + Endpoint:DELETE /shipping/{id}
  + Response: 204 No Content