# Marketplace Technical Foundation

## AVION

## Hackathon Day 2: Planning the Technical Foundation

## 1. Define Technical Requirements

**Frontend Requirements**

- **Pages to Include**:
    1. **Home Page**: Displays an overview of categories and featured products.
    2. **Product Listing Page**: Allows users to browse products by category or search.
    3. **Product Details Page**: Displays detailed information about a product, including customization options (size, color, etc.).
    4. **Cart Page**: Lists selected items with quantity adjustments and the total price.
    5. **Checkout Page**: Collects user information (address, payment) for completing the purchase.
    6. **Order Confirmation Page**: Confirms the successful placement of an order.

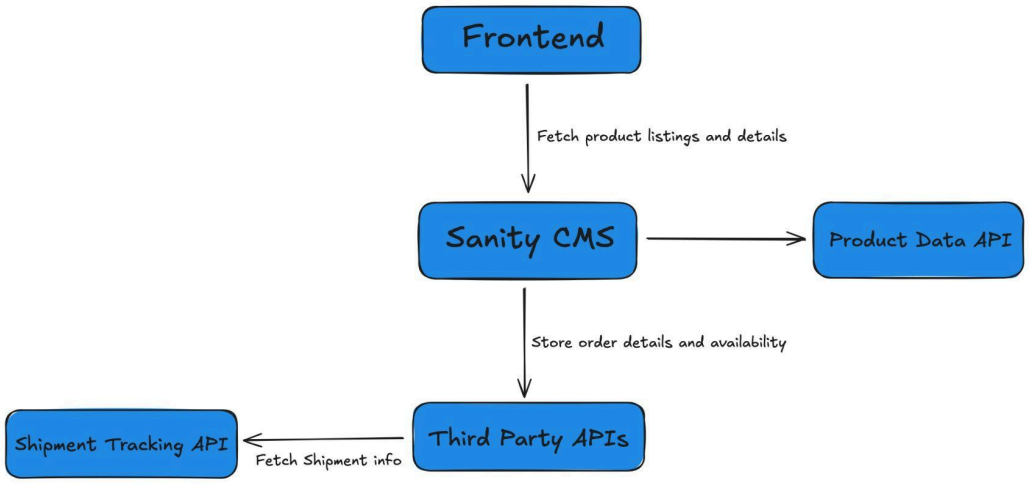
**User Experience**:

- Ensure a **user-friendly interface** for browsing and selecting products.
- Use **responsive design** to provide a seamless experience across mobile and desktop devices.

**Third-Party APIs**

- **Shipment Tracking API**:
    - Integrate an API to fetch real-time delivery updates.

## 2. Design System Architecture:




## 3. System Components and Workflow

1. **User Signup/Login**

    - **Input: User credentials (email, password).**
    - **Database: Sanity stores user data securely.**
    - **API Endpoint: POST /register, POST /login, GET /verify-route for handling authentication.**
    - **Outcome: JWT token issued for session management.**

2. **Content Management (Sanity CMS)**

- **Admin Role: Manages product listings, banners, and blog content.**
- **API Integration: Queries to fetch dynamic content for the frontend.**
- **Outcome: Content rendered seamlessly on the Next.js frontend.**

3. **Product Browsing and Checkout**

- **Database: Sanity CMS stores product details (name, price, stock, description, sizes, etc.).**

### 4. Order Management

- **Database: Sanity CMS stores order data (customer ID, product ID, quantity, status).**
- **API Endpoint: POST /orders to create orders.**

# 4. End Points:

```
const apiEndpoints: ApiEndpoint[] = [

  { endpoint: "/products", method: "GET", purpose: "Fetches all product details", responseExample: JSON.stringify(products) },

  { endpoint: "/products/{id}", method: "GET", purpose: "Fetches details of a specific product by its ID", responseExample: JSON.stringify(products[0]) },

  { endpoint: "/products", method: "POST", purpose: "Adds a new product to the database", responseExample: JSON.stringify({ id: 4, name: "Product D", price: 250 }) },

  { endpoint: "/products/{id}", method: "PUT", purpose: "Updates an existing product by its ID", responseExample: JSON.stringify({ id: 1, name: "Product A Updated", price: 120 }) },

  { endpoint: "/products/{id}", method: "DELETE", purpose: "Deletes a product by its ID", responseExample: JSON.stringify({ message: "Product deleted successfully" }) },

  { endpoint: "/users", method: "GET", purpose: "Fetches all user details", responseExample: JSON.stringify(users) },

  { endpoint: "/users/{id}", method: "GET", purpose: "Fetches details of a specific user by ID", responseExample: JSON.stringify(users[0]) },

  { endpoint: "/orders", method: "GET", purpose: "Fetches all orders", responseExample: JSON.stringify(orders) },

  { endpoint: "/orders/{id}", method: "GET", purpose: "Fetches details of a specific order by ID", responseExample: JSON.stringify(orders[0]) },
```

## Conclusion:

**The Marketplace Technical Foundation for AVION outlines the steps to build a solid, easy-to-use online store. By focusing on frontend design, system structure, and workflows, this plan will help create a smooth experience for both customers and store managers.**

**We've planned out important features like user, content management with Sanity CMS, and product browsing with a simple checkout process. We've also included an integration with a shipment tracking API to give real-time updates to users, making the shopping experience even better.**