MARKETPLACE TECHNICAL FOUNDATION

MARKETPLACE BUILDER HACKATHON 2025 DAY 2

Technical Planning:

This high-level technical plan defines the foundational architecture, core technologies, and essential workflows that are critical to the successful development of my e-commerce platform dedicated to trendy, eco-friendly men's T-shirts and Men's and Women's clothing

---

1. Architecture Overview

- **Frontend**:
  - **Framework**: Next.js (React-based for Server-Side Rendering and Static Site Generation).
  - **Styling**: Tailwind CSS for responsiveness and faster development.
- **Backend**:
  - **CMS**: Sanity CMS for managing dynamic content (products, orders, and customers).
  - **API**: Custom Next.js API routes for business logic.
- **Hosting & Deployment**:
  - **Platform**: Vercel for seamless hosting and automatic CI/CD integration.

---

2. Key Components

Frontend Requirement

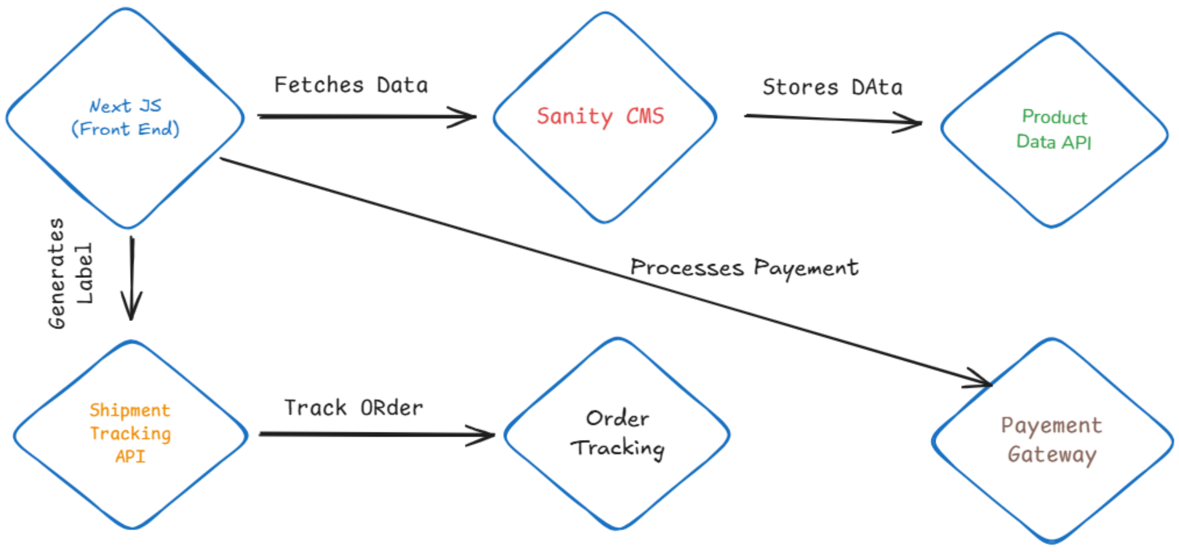
- **Pages**:
  - **Home**: Featured collections, banners, and promotions.
  - **Product Listing**: Filter and sort products by size, color, and price.
  - **Product Details**: Display product details, images, and customization options.
  - **Cart**: Manage selected items with pricing, size, and quantity.
  - **Checkout**: Input shipping and payment details securely.
  - **Order Confirmation**: Display a summary of the order and tracking details.
- **Key Features**:
  - Responsive design for all devices (mobile-first approach).
  - Interactive UI/UX using React hooks and state management.
  - SEO optimizations for organic traffic.

Sanity CMS as Backend

- **Sanity CMS**:
  - **Schemas**:
    - **Product Schema**: `name`, `description`, `images`, `price`, `sizes`, `stock`.
    - **Customer Schema**: `name`, `email`, `phone`, `address`.
    - **Order Schema**: `orderId`, `productList`, `totalAmount`, `status`.
- **Third-Party Integrations**:
  - **Payment Gateway**: Bank/EasyPaisa for secure payments.
  - **Shipping API**: Leopard Courier for real-time shipment tracking.
  - **Email Notifications**: Twilio/SendGrid for order updates.

---

2. Design System Architecture



2a. Example System Architecture:

Starting Point:

- The system begins with a decision node where the user can either proceed as a **Customer** or an **Admin**.
  - **Customer**: Regular users who interact with the platform for browsing and placing orders.
  - **Admin**: Backend managers who oversee product management and reporting.

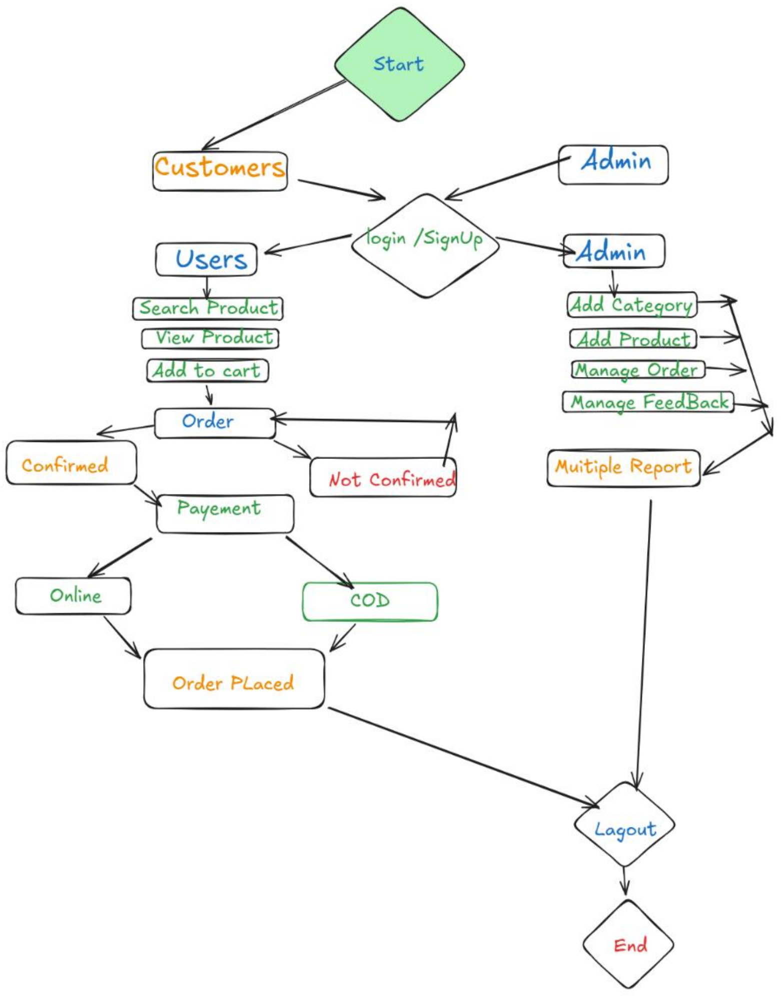
Customer Workflow

1. **Login or Signup**
   - Users must log in or register an account to proceed.
2. **Customer Actions**
   - **Search Product**: Customers search for items using filters or categories.
   - **View Product**: Details of selected products are displayed.
   - **Add to Cart**: Products can be added to the cart for purchase.
3. **Order Confirmation**
   - Users proceed to the checkout. If confirmed, they can move forward. If not, they can make changes.
4. **Payment Options**
   - Two payment methods are available:
     - **Online Payment**: Users complete transactions digitally.
     - **Cash on Delivery (COD)**: Payment is made upon receiving the order.
5. **Order Placed**
   - Once payment is complete, the order is marked as placed.
6. **Logout**
   - Users can log out after completing their actions.

Admin Workflow

1. **Login or Signup**
   - Admins must authenticate themselves to access the platform.
2. **Admin Actions**
   - **Add Category**: Create or modify product categories.
   - **Add Product**: Add or update products in the system.
   - **Assign Orders**: Oversee order fulfillment and delivery assignment.
   - **Manage Reports**: View and generate multiple reports related to sales, inventory, and user activities.
3. **Logout**
   - Admins can log out after completing management tasks.

Workflow Diagram



2. Plan API Requirements:

1. Fetch Products

- **Endpoint Name**: `/products`
- **Method**: GET
- **Description**: Retrieve all available products from the database, including details like ID, name, price, stock, and images.

2. Create Order

- **Endpoint Name**: `/orders`
- **Method**: POST
- **Description**: Place a new order by submitting customer details, selected products, and payment status.

3. Track Shipment

- **Endpoint Name**: `/shipment`
- **Method**: GET
- **Description**: Get real-time updates on shipment status using third-party logistics APIs.

4. Add New Product

- **Endpoint Name**: `/products`
- **Method**: POST
- **Description**: Add a new product to the store catalog, including details like name, price, stock, and image URL.

5. Fetch Customer Details

- **Endpoint Name**: `/customers/{customerId}`
- **Method**: GET
- **Description**: Retrieve specific customer information using their unique I

6. Update Order Status

- **Endpoint Name**: `/orders/{orderId}/status`
- **Method**: PATCH
- **Description**: Modify the status of an order, such as updating it from "Processing" to "Shipped."

Prepared By: Usman Naseem

Slots: Sunday 2 to 5