Marketplace Hackathon Technical Framework

1. Website Layout and Sections

The platform consists of the following main sections:

a. **Homepage**

- Serves as the introductory page, showcasing the platform's goal and highlighting key features and offerings.
- Content dynamically retrieved and managed through Sanity CMS.

b. **About Us**

- A section dedicated to describing the purpose, values, and background of BANDAGE.
- Fully customizable via Sanity CMS for easy updates.

c. **Contact Us**

- Includes a form enabling visitors to communicate directly with the team.
- Submitted forms can connect to backend systems for processing.

d. **Product Listings**

- Displays a catalog of available items with pricing and short descriptions.
- Data is fetched dynamically using the Sanity CMS.

e. **Product Details**

- Provides comprehensive information for each item, including specifications, images, reviews, and pricing.
- Data dynamically sourced from Sanity based on the selected product.

f. **Shopping Cart**

- Displays the user's selected products for purchase.
- Enables real-time management, such as adding, removing, or updating product quantities.

g. **User Authentication**

- Managed via Clerk, offering secure login and signup features.
- Options include social logins and email/password authentication.

h. **Payment and Checkout**

- Integrated with Stripe for smooth and secure payment processes.
- Supports multiple payment options.

i. **Order Shipping and Tracking**

- Enables users to track deliveries and view shipping details.
- APIs like Shippo and EasyPost offer real-time tracking and updates.

j. **Utility Pages**

- Additional pages include error handling (e.g., 404 pages) and loading screens to enhance the user experience.

---

2. Backend Powered by Sanity CMS

- Sanity serves as the core content management system to control dynamic elements across the site.
- Products, product information, banners, and homepage content can be updated without requiring code modifications.

3. Clerk for User Authentication

- Clerk manages secure user login and registration.
- Key features include:
    - Safe handling of user credentials.
    - Support for password recovery, two-factor authentication, and social logins.
- Easily integrates with Next.js for real-time user session management.

4. Stripe for Payments

- Stripe API powers secure and seamless payment functionality.
- Highlights:
    - Supports multiple payment methods (e.g., credit cards, digital wallets).
    - Manages successful and failed transactions for a smooth checkout process.

5. Shipping Management with APIs

- **Shippo** and **EasyPost** handle shipment operations and tracking.
    - Shippo: Manages shipping labels and carrier accounts.
    - EasyPost: Provides live tracking updates and delivery status.

---

6. Technology Stack

a. Next.js

Designed to create a scalable, SEO-optimized, and high-performance website.
● Offers server-side rendering (SSR) and static site generation (SSG) to enhance speed and efficiency.

- Enables building a fast, SEO-optimized website.
- Offers server-side rendering (SSR) and static site generation (SSG).

b. Sanity CMS

- Acts as the headless CMS for managing content like products and banners.

c. Clerk

- Handles authentication and user management for secure and modern login experiences.

d. Stripe

- Manages payment transactions during checkout.

e. Shippo and EasyPost

- Simplify shipment processes and provide real-time order tracking.

---

7. Enhanced User Experience

- **Dynamic Product Content:** Information such as product specs and prices is fetched and displayed dynamically from Sanity CMS.
- **Seamless Authentication:** Clerk provides a user-friendly login/signup experience.
- **Efficient Checkout:** Stripe integration ensures secure payment processes.
- **Real-Time Tracking:** Shipping APIs enable detailed tracking information post-purchase.

8. Scalability and Maintenance

- Designed to scale alongside business growth with tools like Sanity, Clerk, Stripe, Shippo, and EasyPost.
- Admins can update the website's content easily without altering code.
- API integrations minimize ongoing maintenance for core functionalities.

---

9. User Journey on the Website

Step 1: Visiting the Platform

- **Frontend (Next.js):**
    - Users land on the homepage.
    - Featured products and banners are dynamically displayed using Sanity CMS.

- If logged in, users are greeted with personalized information managed by Clerk.

Step 2: User Authentication

- **Login/Signup via Clerk:**
  - Supports email/password and social logins (e.g., Google, GitHub).
  - Securely manages user sessions and authentication data.
  - Logged-in users gain access to:
    - Profile management.
    - Viewing previous orders.
    - Updating account settings.

Step 3: Browsing Products

- **Products Page:**
  - Users view a list of products fetched dynamically from Sanity CMS.
  - Future upgrades could include a wishlist feature for logged-in users.

Step 4: Viewing Product Details

- **Product Details Page:**
  - Displays detailed information for a selected product, including stock availability and reviews.
  - Data is fetched dynamically via API calls to Sanity CMS.

Step 5: Adding to Cart

- **Shopping Cart:**
  - Clicking "Add to Cart" temporarily saves the item in the user's session or app state.
  - Unauthenticated users are prompted to log in before proceeding.
  - Once logged in, the cart syncs with the user's profile.

Step 6: Checkout

- **Payment Processing:**
  - Stripe calculates the cart's total value.
  - Redirects users to a secure payment page for checkout.

Step 7: Payment Confirmation

- **Stripe API:**

- Processes payment and sends confirmation to the platform.
- Order details are saved in the Sanity database for record-keeping.

Step 8: Shipment Tracking

- **Shipping APIs:**
  - Generates tracking numbers and provides real-time updates for deliveries.
  - Users can view order and shipping status on their profile page.

Step 9: Profile and Order History

- **Profile Page:**
  - Logged-in users can view past orders, update personal details, and track active shipments.
  - Order data is dynamically displayed from Sanity CMS.

---

10. Key API Endpoints

Endpoint	Method	Purpose	Response Example
/products	GET	Retrieves all available products	{ "id": 1, "name": "Product X", "price": 150 }
/orders	POST	Creates a new order	{ "orderId": 987, "status": "Order Placed" }
/order-status	GET	Fetches order status	{ "orderId": 987, "status": "Processing" }
/shipment-status	GET	Tracks shipment status via shipping APIs	{ "shipmentId": 456, "status": "Dispatched" }
/user/cart	GET	Retrieves current cart items	{ "cartItems": [{ "id": 1, "quantity": 2 }]}
/user/cart/add	POST	Adds a product to the cart	{ "message": "Product added successfully" }
/user/cart/remove	DELETE	Removes a product from the cart	{ "message": "Product removed successfully" }

---

11. Summary of Features

- **Core Pages:**

- **Homepage:** Overview of offerings, dynamically updated.
- **About Us:** Company background and purpose.
- **Contact:** Easy communication via forms.
- **Products:** Comprehensive product catalog.
- **Details:** In-depth product information.
- **Cart:** Real-time management of selected items.
- **Authentication:** Secure login/signup powered by Clerk.
- **Checkout:** Stripe-enabled payment process.
- **Shipping:** Live tracking using Shippo and EasyPost.

- **Backend Capabilities:**
  - **Sanity CMS:** Dynamic content management.
  - **APIs:** Streamlined authentication, payment, and shipping workflows.

---

10. Key API Endpoints

Endpoint	Method	Purpose	Response Example
/products	GET	Retrieves all available products	{ "id": 1, "name": "Product X", "price": 350 }
/orders	POST	Creates a new order	{ "orderId": 112, "status": "Order Placed" }
/order-status	GET	Fetches order status	{ "orderId": 982, "status": "Processing" }
/shipment-status	GET	Tracks shipment status via shipping APIs	{ "shipmentId": 345, "status": "Dispatched" }
/user/cart	GET	Retrieves current cart items	{ "cartItems": [{ "id": 1, "quantity": 2 }]}
/user/cart/add	POST	Adds a product to the cart	{ "message": "Product added successfully" }
/user/cart/remove	DELETE	Removes a product from the cart	{ "message": "Product removed successfully" }