MARKETPLACE HACKHATHON TECHNICAL ARCHITECTURE:-

1. Website Structure and Pages

The website includes the following core pages:

a. Home

- The landing page provides an overview of the website's purpose and highlights key products and features.
- Dynamic content displayed using **Sanity** CMS.

b. About

- A page dedicated to detailing the mission, vision, and background of BANDAGE.
- Content managed via **Sanity**, ensuring easy updates and customizability.

c. Contact

- A contact form for users to get in touch with the team.
- Form submissions can be integrated into backend services for further processing.

d. Products

- Displays a list of products, their prices, and brief details.
- Products and data are fetched dynamically from **Sanity**.

e. Product Details

- A detailed view for each product, showing specifications, images, reviews, and price.
- Fetched dynamically from **Sanity** based on the product selected.

f. Cart

- A user's selected products for purchase are displayed in the cart.
- Data managed dynamically, allowing users to add, remove, or update product quantities.

g. Login and Signup

- Authentication and user management handled via **Clerk**.
- Secure login and signup options with features like social login or email/password-based authentication.

h. Checkout and Payment

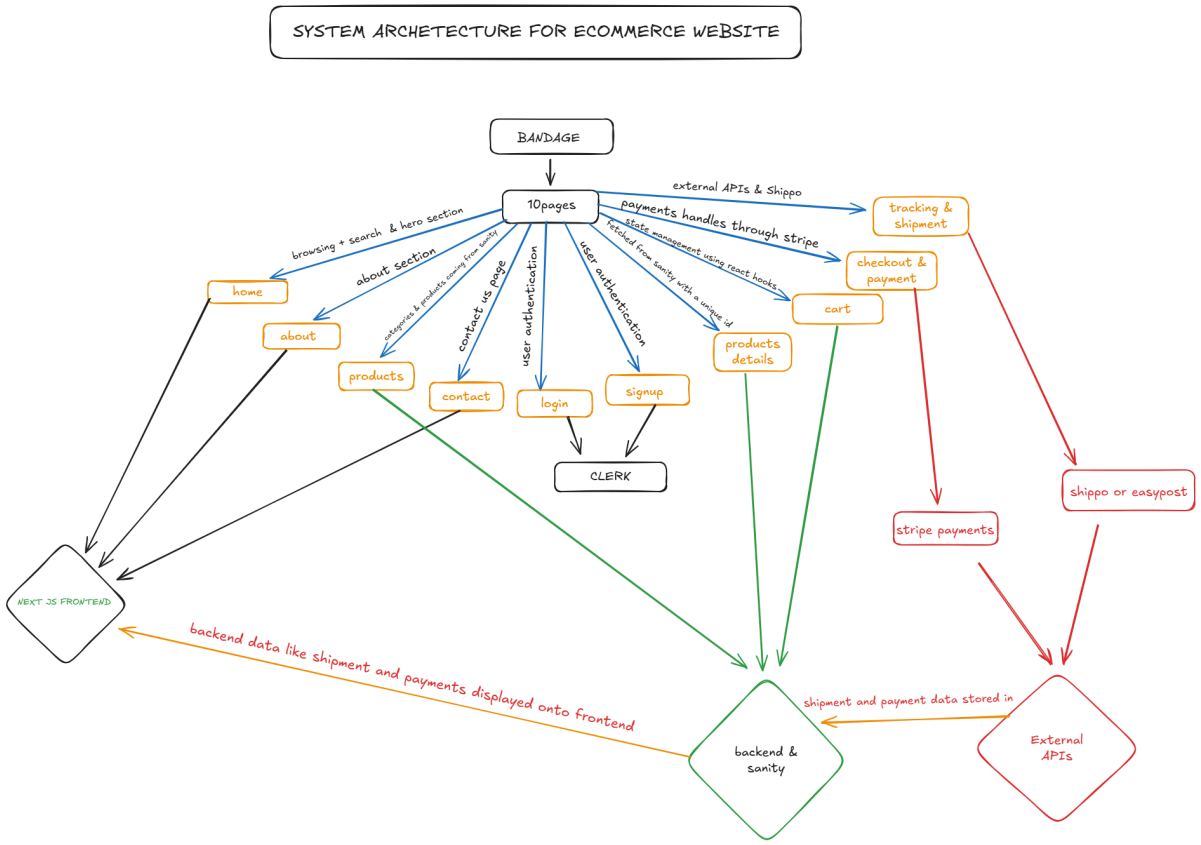
- Checkout functionality is seamlessly integrated with **Stripe** for secure payment processing.
- Supports multiple payment methods.

i. Shipment and Tracking

- Users can track their orders and view shipping details.
- Integration with **Shippo** and **EasyPost** APIs ensures real-time updates and tracking features.

j. Other Utility Pages

- Includes error handling pages (e.g., 404 page) and loading pages to enhance user experience.



2. Backend Integration with Sanity

- **Sanity** serves as the central CMS for managing and displaying dynamic content across the website.
- Products, product details, and other content like banners and text for the home page are fetched dynamically.
- This allows admins to update the website's content without modifying the code.

---

3. Authentication Using Clerk

- **Clerk** handles user authentication for the login and signup pages.
- Features:
    - Secure user data handling.
    - Password recovery, multi-factor authentication, and social login options.
- Provides seamless integration with Next.js for real-time authentication.

---

4. Payment Integration with Stripe

- The **Stripe API** handles secure payment processing.
- Features include:
    - Multiple payment options like credit cards, digital wallets, etc.
    - Payment success and failure management, ensuring a seamless checkout process.

---

5. Shipment and Tracking with Shippo and EasyPost

- **Shippo** and **EasyPost** APIs are used for shipment management and tracking.
- **Shippo** handles:
    - Generating shipping labels.
    - Managing carrier accounts and rates.
- **EasyPost**:
    - Provides real-time tracking updates for shipments.
    - Allows users to view the status and location of their packages.

---

6. Technology Stack

a. Next.js

- Used for building a scalable, SEO-friendly, and fast-loading website.
- Provides server-side rendering (SSR) and static site generation (SSG) for improved performance.

b. Sanity

- Acts as the headless CMS for managing dynamic content, such as product data and other website elements.

c. Clerk

- Used for user authentication and user management, ensuring secure and modern login/signup experiences.

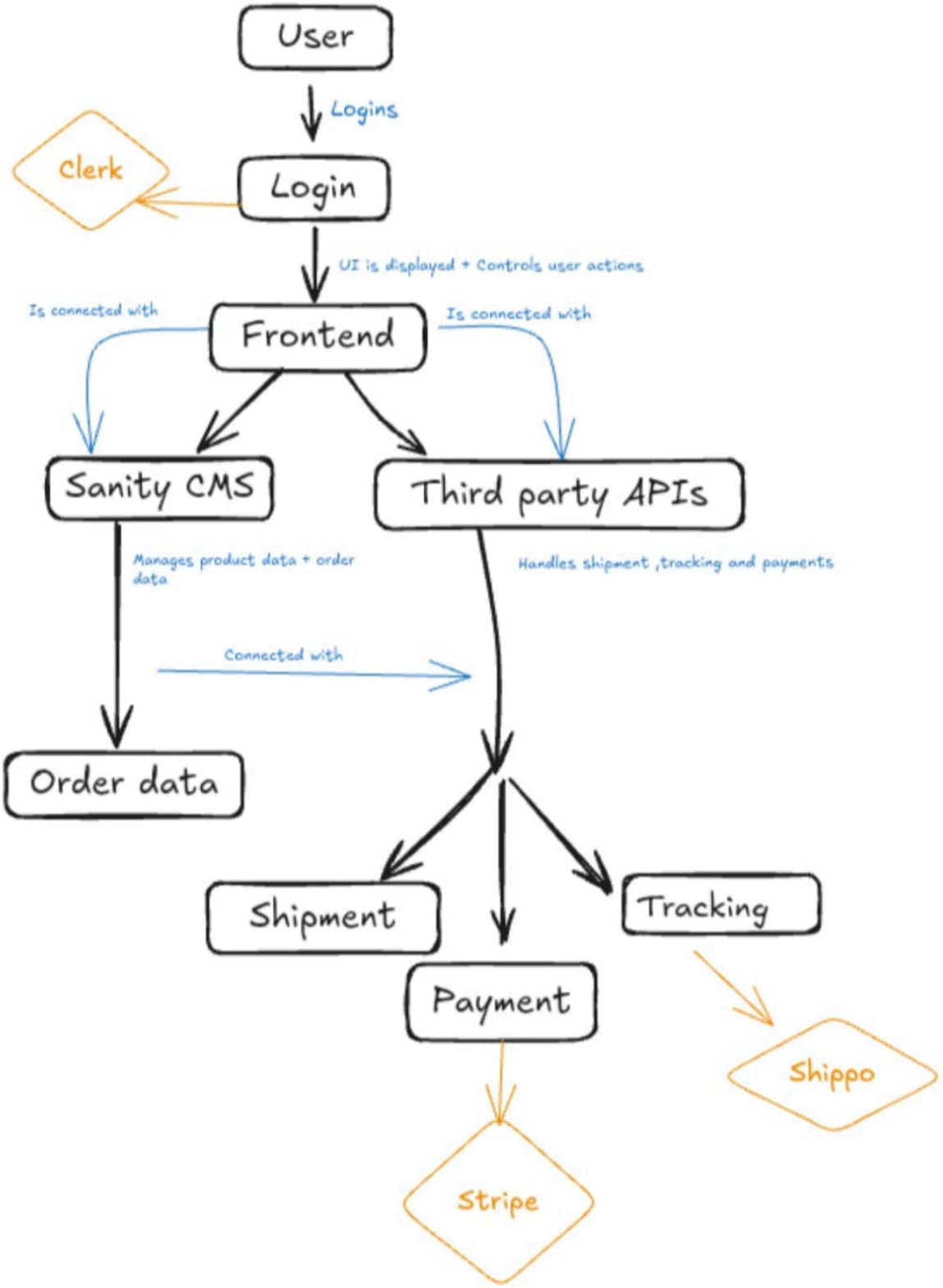
d. Stripe

- Handles secure payment processing during the checkout phase.

e. Shippo and EasyPost

- These APIs provide an efficient way to manage shipments and allow users to track their orders in real-time.

SYSTEM ARCHITECTURE



---

7. User Experience Highlights

- **Dynamic Content**: Pages like Products and Product Details dynamically fetch and display information based on the Sanity CMS.
- **Secure and User-Friendly Authentication**: Clerk ensures a seamless login/signup process.
- **Efficient Payments**: Stripe integration allows users to pay securely and effortlessly.
- **Real-Time Shipment Tracking**: Shippo and EasyPost enhance the post-purchase experience by providing detailed tracking information.

---

8. Scalability and Maintenance

- With the integration of **Sanity**, **Clerk**, **Stripe**, **Shippo**, and **EasyPost**, the BANDAGE website is designed to scale easily as the business grows.
- Content can be updated dynamically, and external APIs ensure minimal maintenance requirements for key features like authentication, payment, and shipping.

—----------

9. Steps Involved in my website

Step 1: User Visits the Website

Frontend (Next.js):

- **The user lands on the Home Page.**
- **Dynamic content, such as featured products and banners, is fetched from Sanity CMS via API calls and displayed dynamically.**
- **If the user is already logged in, their session is managed securely by Clerk Authentication, and they are welcomed back with personalized details.**

Step 2: User Logs In/Signs Up

Authentication via Clerk:

- **Users can log in or sign up using:**
  - **Email and password.**
  - **Third-party options like Google, GitHub, or other providers supported by Clerk.**
- **Clerk Authentication:**
  - **Manages user sessions and securely stores authentication data.**

    - Ensures only authenticated users can proceed to specific actions (like accessing the cart or checkout).
- Logged-in users can:
    - Access their Profile Page.
    - View their previous orders.
    - Manage account settings.

Step 3: User Browses Products

Products Page:

- The user navigates to the Products Page, which displays a list of products.
- Sanity CMS:
    - The product data, including name, price, images, and descriptions, is stored in the Sanity database.
    - An API fetches this data and displays it dynamically on the page.
- Logged-in users:
    - Can save products to a wishlist or favorites (optional feature for future integration).

Step 4: User Clicks on a Product

Product Details Page:

- The user clicks on a product to view more information.
- Sanity CMS:
    - The specific product's details (e.g., specifications, price, stock availability) are fetched via an API.
    - This data is displayed dynamically to the user on the Product Details Page.

Step 5: User Adds Product to Cart

Frontend (Next.js):

- When the user clicks the Add to Cart button:
    - The product information (e.g., product ID, quantity, price) is temporarily stored in the user's session or managed using a state management system like Redux or Context API.
    - If the user is not logged in:
        - They are prompted to log in or sign up before proceeding to checkout.

        - Once logged in, their cart is saved and synchronized with their account.

Step 6: User Proceeds to Checkout

Checkout Process:

- The user clicks the Checkout button to review the order details and proceed to payment.
- Stripe Payment Integration:
    - The total cart value is calculated.
    - The user is redirected to a secure payment page powered by Stripe, where they can enter their payment details (e.g., credit card).

Step 7: Payment Confirmation

Stripe API Integration:

- Once the payment is successful:
    - Stripe sends a payment confirmation response back to the website via its API.
    - Sanity CMS:
        - The order details (e.g., user information, purchased products, payment status) are stored in the Sanity database for tracking and future reference.

Step 8: Shipping and Order Tracking

Third-Party APIs (Shippo/EasyPost):

- The order is passed to shipping APIs like Shippo or EasyPost.
- These APIs:
    - Generate a tracking number for the shipment.
    - Update the order's shipping status.
- The user can view the order status and tracking details on the Order Tracking Page.

Step 9: User Views Profile/Order History

Profile Page:

- **Logged-in users can access their Profile Page to:**
  - **View their order history.**
  - **Track the status of their orders.**
  - **Update personal information like addresses or contact details.**
- **Sanity CMS:**
  - **The order history data is fetched dynamically from the Sanity database and displayed on the Profile Page.**

10. API Endpoints

Endpoint	*Method*	*Purpose*	*Response Example*
`/products`	*GET*	*Retrieves a list of all available products*	`{ "id": 1, "name": "Product X", "price": 150 }`
`/orders`	*POST*	*Creates a new order and saves it in the database*	`{ "orderId": 987, "status": "Order Placed" }`
`/order-status`	*GET*	*Fetches the status of a specific order*	`{ "orderId": 987, "status": "Processing", "ETA": "3 days" }`
`/shipment-status`	*GET*	*Tracks shipment status through third-party shipping API*	`{ "shipmentId": 456, "status": "Dispatched", "location": "Hub A" }`
`/express-shipment`	*GET*	*Retrieves real-time status for express deliveries*	`{ "orderId": 321, "status": "Out for Delivery", "ETA": "30 mins" }`
`/user/cart`	*GET*	*Retrieves items currently in the user's cart*	`{ "cartItems": [ { "id": 1, "name": "Product X", "quantity": 2 } ], "totalPrice": 300 }`
`/user/cart/add`	*POST*	*Adds a new product to the cart*	`{ "message": "Product added successfully", "cartItems": [ { "id": 1, "name": "Product X", "quantity": 3 } ] }`

`/user/cart/remove`	*DELETE*	*Removes a product from the cart*	`{ "message": "Product removed successfully", "cartItems": [ { "id": 2, "name": "Product Y", "quantity": 1 } ] }`
`/reviews`	*POST*	*Submits a review for a specific product*	`{ "productId": 1, "reviewId": 765, "message": "Review submitted successfully" }`
`/track/order`	*GET*	*Retrieves complete tracking details for an order*	`{ "orderId": 987, "status": "Shipped", "lastLocation": "City Hub", "ETA": "2 days", "trackingDetails": [ { "date": "2025-01-16", "status": "Dispatch" } ] }`

---

11. Short Overview

🔑 **Core Pages**:

- **Home** 🏠: Overview of products and features, dynamic content via Sanity CMS.
- **About** 📖: Mission, vision, and background details of BANDAGE.
- **Contact** 📞: Contact form for user inquiries.
- **Products** 🛍️: List of products with prices and brief details.
- **Product Details** 🛒: In-depth specs, images, reviews, and price.
- **Cart** 🛒: Manage selected products for checkout.
- **Login/Signup** 🔑: Authentication using Clerk (social login options).
- **Checkout & Payment** 💳: Secure Stripe payment integration.
- **Shipment & Tracking** 🚚: Real-time order tracking with Shippo and EasyPost.

🔧 **Backend Integration**:

- **Sanity CMS** 🌐: Dynamic content management for products and pages.
- **Clerk Authentication** 🔐: Secure login and user management.
- **Stripe Payment** 💳: Multiple payment methods for smooth checkout.

- **Shippo & EasyPost** 📦: Shipment tracking and label generation.

⚙️ **Technology Stack**:

- **Next.js** 🚀: Fast, SEO-friendly frontend framework.
- **Sanity** 📦: Headless CMS for content management.
- **Clerk** 🔑: User authentication.
- **Stripe** 💳: Payment processing.
- **Shippo/EasyPost** 🚚: Shipment tracking.

💡 **User Experience**:

- **Dynamic Content** 📈: Product pages and details fetched from Sanity.
- **Seamless Authentication** 🔐: Easy login/signup with Clerk.
- **Effortless Payments** 💳: Secure payments via Stripe.
- **Real-Time Tracking** 🚚: Track orders with Shippo/EasyPost.

📈 **Scalability & Maintenance**:

- Easily scalable as business grows.
- Simple updates via Sanity CMS and minimal maintenance with external APIs.

Thanks to **Sir Ameen Alam** 🙏 for the clear guidance and helping us implement these features successfully! 🎉