Hackathon # 3 (Day 02)

Foundation For "StylistaBuy"

System Architecture

➢ Frontend

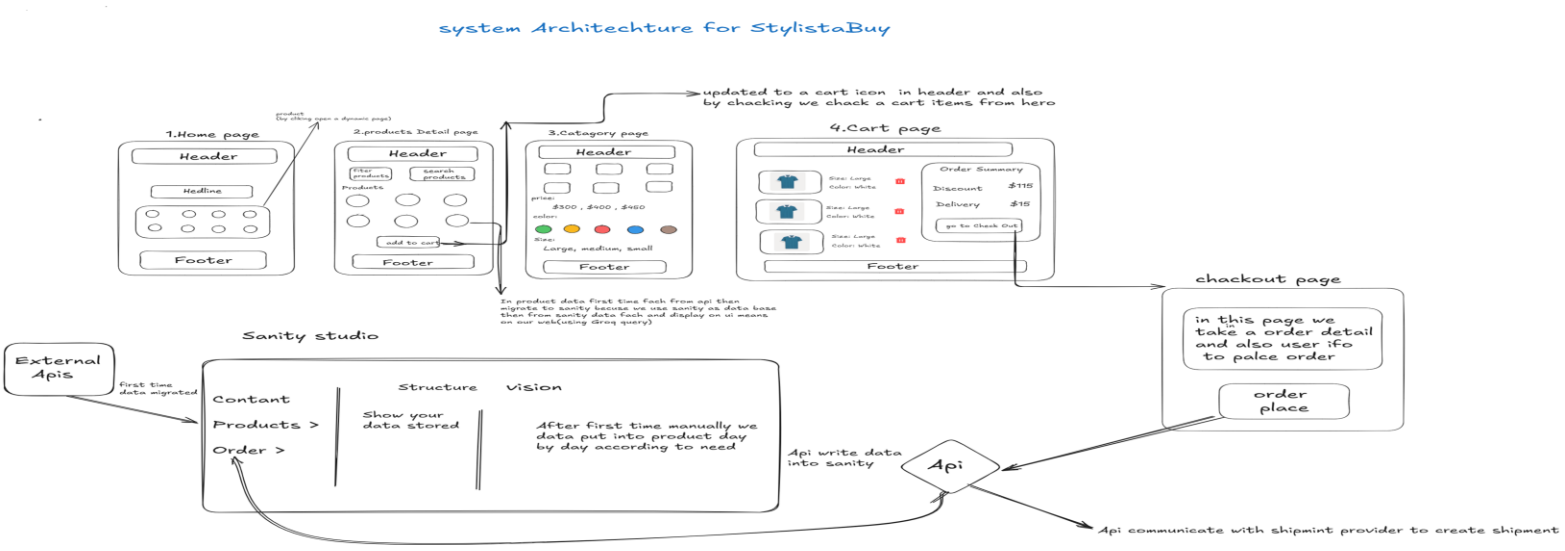
- **Framework**: Next.js.
- **Features**:
  - Responsive design using Tailwind CSS or Bootstrap.
  - Pages:
    1. **Home Page**: Banner, recent products, and product categories.
    2. **Product Page**: Displays all products with search and filters.
    3. **Contact Page**: Form for inquiries and information requests.
    4. **Catagory**: Displays different catagorys cloths (e.g(shuts, pants, geens etc)).
  - State Management:Redux Toolkit for managing cart, user session, and UI state.

➢ Backend (Headless CMS + APIs)

1. **Sanity**:
   - Stores product data, blog content, and orders.
   - Acts as the primary data source for the frontend.
2. **Custom APIs**:
   - **Data Migration API**: Fetches product data from external API and migrates it to Sanity.
   - **Order Placement API**: Captures orders and writes them to Sanity.
3. **Third-Party APIs**:
   - Shipment and tracking APIs for real-time order updates.

Database

- **Sanity Database**:
  - Stores structured content for products, blogs, and orders.
  - Provides CMS capabilities for managing content.



➢ Files Name

- **Home Page** – Featured products, categories, and promotions
- **Product Listing Page** – Grid/list view of dresses with filters
- **Product Detail Page** – Dress details, images, price, and "Add to Cart"
- **Shopping Cart** – Items, quantity, subtotal, and checkout button
- **Checkout Page** – Billing, shipping, and payment details

• Workflow:

1. Add to Cart Workflow

User Action:

- A user clicks "Add to Cart" on the product page.

Frontend (Client-Side):

- The product details (ID, name, price, quantity) are stored in the client state (e.g., React Context or Redux).
- The cart state is updated to show the added item.

2. Order Placement Workflow

User Action:

- User proceeds to checkout from the cart page.
- Enters delivery address, selects a payment method, and confirms the order.

❖ **Frontend (Client-Side)**:

- Sends the cart data, user details, and payment information to the **Order Placement API**.

Backend:

- The **Order Placement API** validates the order data.
- Writes the order details to Sanity under the orders collection.
- Calls the third-party shipment API to generate a shipment and fetch a tracking number.

Response:

- The API returns the order confirmation with tracking details.
- Frontend updates the order confirmation page for the user.

3. API Handling Workflow

A. Product Data Migration

1. **External API Fetch**:
   - A cron job or manual trigger fetches product data from an external API.
2. **Data Transformation**:
   - The data is cleaned and formatted to match Sanity's schema.
3. **Sanity Integration**:
   a. The transformed data is sent to Sanity via its Content API to populate the products collection.

B. Fetching Data from Sanity

1. **Frontend Request**:
   - Next.js fetches product data from Sanity using GROQ queries or the GraphQL API.
2. **Caching**:
   - Data is cached at the server level for faster subsequent loads (e.g., using ISR in Next.js).

C. Order Placement API

1. **Request**:
   - The frontend sends order details to the custom Order Placement API.
2. **Sanity Write**:
   - The API writes the order data into Sanity for record-keeping.
3. **Third-Party Shipment API**:
   - The API communicates with a shipment provider to create a shipment.

➢ System Overview

❖ Frontend

- Home Page
- Product Page
- Contact Page
- Blog Page

- **Backend**

- Sanity for Products, Orders, Blogs
- External APIs for Initial Product Data
- Order Placement API

- **Third-Party APIs**

- Payment Gateway
- Shipment Tracking