Building the Marketplace Foundation Hackathon-3 (day-2)

Prepared By: Nadia Shaikh

Roll No: 00415874

BY Nadia Shaikh

QUARTER 2 GIAIC

Marketplace Technical Foundation Hacto E-commerce

1. Frontend Requirements:

- **User Interface (UI):**

  A user-friendly interface has been developed for browsing furniture products on the **Hacto** E-commerce website.
  The design is clean, modern, and aesthetically pleasing to enhance user experience.

- **Responsive Design:**

  The **Hecto** E-commerce website is fully responsive, ensuring it works seamlessly on both mobile and desktop devices.

Essential Pages:

- **Home Page:**

  The main page of **Hecto** E-commerce, showcasing featured products, promotions, and including a search bar for easy navigation.

- **Product Listing Page:**

  A page displaying a list of furniture products with filters for categories, price, and popularity, making it easy for users to find what they are looking for.

- **Product Details Page:**

  A detailed view of a single product, including images, description, price, and customer reviews (to be added later).

- **Shopping Cart:**

  The functionality for adding items to the cart has been implemented. This page displays selected products, their quantities, prices, and provides the option to remove items.

- **Checkout Page (to be added):**

  This page will be developed in the coming days, allowing users to enter their details, choose shipping options, and make payments.

- **Order Confirmation Page (to be added):**
  This page will be developed in the coming days to confirm the user's order with an order summary and estimated delivery date.

Additional Features:

- **Search Bar:**
  A search functionality has been integrated into the **Hecto** website, allowing users to quickly find products.

- **Navigation Menu:**
  Easy navigation has been provided to different categories and sections of the **Hecto** website, ensuring a seamless browsing experience.

- **User Account (to be added):**
  Login and sign-up functionality along with account management features will be provided later.

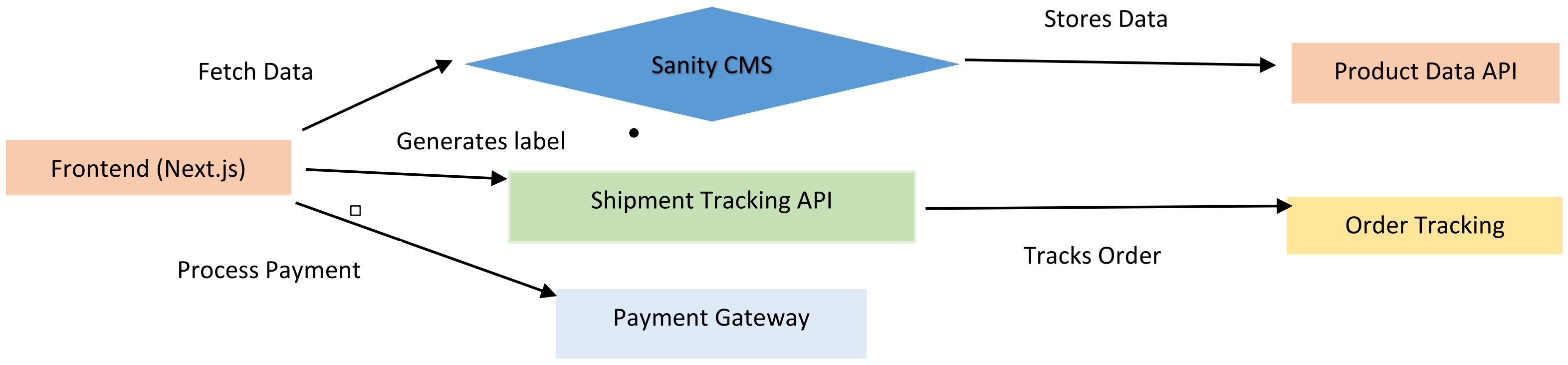
- **Wish list (to be added):**
  There are plans to provide users the option to save products to a wish list for future reference.

- **Customer Reviews (to be added):**
  A section will be provided where users can read and write reviews on products.

2. Design System Architecture:

- **Introduction:**
  The system architecture diagram below outlines the interactions and data flow between different components of the Hecto e-commerce website. This architecture ensures a scalable, efficient, and user-friendly platform for managing furniture products, handling orders, and processing payments.

  - **System Architecture Diagram:**
  -



Component Descriptions:

Frontend (Next.js):

Reason for Use: Next.js is used for its powerful server-side rendering capabilities, which improve the performance and SEO of the website. It provides a seamless user experience with fast page loads and dynamic content rendering.

Function: It serves as the user interface layer where users interact with the website, browse products, manage their cart, and proceed to checkout.

Sanity CMS:

Reason for Use: Sanity CMS is chosen for its flexibility and real-time content management capabilities. It allows for easy management of product data and orders through a user-friendly interface.

Function: It manages and stores product and order data. Sanity CMS connects to the Product Data API to retrieve product information and update the frontend with the latest data.

Product Data API:

Reason for Use: The Product Data API provides a structured way to access detailed product information. It ensures that the data is consistent and up-to-date across the system.

Function: It provides detailed information about the products to Sanity CMS, which is then used to display product details on the website.

Third-Party API (Shipment Tracking):

Reason for Use: The Shipment Tracking API is integrated to provide users with real-time tracking information for their orders. It enhances the user experience by keeping them informed about their shipment status.

Function: It generates shipping labels and tracks orders, providing tracking information to the frontend.

Payment Gateway:

Reason for Use: A secure payment gateway is essential for processing payments efficiently and safely. It handles payment transactions and ensures that sensitive payment information is processed securely.

Function: It processes payments securely, interacting with the frontend to handle payment details and confirm successful transactions.

Workflow Description:

In this architecture, a typical data flow could look like this:

A user visits the marketplace frontend to browse products.

The frontend makes a request to the Product Data API (powered by Sanity CMS) to fetch product listings and details, which are displayed dynamically on the site.

When the user places an order, the order details are sent to Sanity CMS via an API request, where the order is recorded.

Shipment tracking information is fetched through a Third-Party API and displayed to the user in real time.

Payment details are securely processed through the Payment Gateway, and a confirmation is sent back to the user and recorded in Sanity CMS.

Key Workflows to Include:

User Registration:

User signs up -> Data is stored in Sanity -> Confirmation sent to the user.

Product Browsing:

User views product categories -> Sanity API fetches data -> Products displayed on the frontend.

Order Placement:

User adds items to the cart -> Proceeds to checkout -> Order details saved in Sanity.

Shipment Tracking:

Order status updates fetched via 3rd-party API -> Displayed to the user.

3. Plan API Requirements:

Introduction:

This section defines the API endpoints needed based on the data schema for the **Hacto** e-commerce website. Each endpoint is described with its method, purpose, payload, and response.

API Endpoints:

1. Endpoint Name: /products

Method: GET

Description: Fetch all available products from Sanity.

Response Example:

```
{
  "Id": 1,
  "name": "Product A",
  "price": 100,
  "stock": 20,
  "image": "url_to_image"}
```

2. Endpoint Name: `/orders`
Method: `POST`
Description: Create a new order in Sanity.

Payload Example:

```
{
  "customer Info": {
    "name": "John Doe",
    "email": "john@example.com",
    "address": "123 Main St"
  },
  "productDetails": [
    {
      "productId": 1,
      "quantity": 2
    }
  ],
  "paymentStatus": "Paid"
}
```

Response Example:

```
{
  "orderId": 123,
  "status": "Success"
}
```

3. Endpoint Name: `/shipment`
Method: `GET`
Description: Track order status via third-party API.

Response Example:

```
{

  "shipmentId": "SHIP123",

  "orderId": 123,

  "status": "In Transit",

  "expectedDeliveryDate": "2025-01-16"

}
```

Conclusion:

This detailed API documentation ensures a clear understanding of the endpoints needed for the Hacto e-commerce website. It outlines the methods, descriptions, payloads, and responses for each endpoint, providing clarity for implementation and ensuring seamless integration with the marketplace workflows.

1. System Architecture Overview:

System Architecture Diagram:

- User [User] --> |Sign in| Clerk [Clerk Authentication]
- User --> |Browses| Frontend [Next.js]
- Frontend --> |Fetches Data| Sanity [Sanity CMS]
- Frontend --> |Manages Cart| Context [use Context API]
- Frontend --> |Places Order| Stripe [Stripe Checkout]
- Sanity --> |Stores| ProductData [Product Data]
- ShipEngine --> |Tracks Order| Order Tracking [Order Tracking]

---

2. Key Workflows:

User Authentication (Clerk):

- Use Clerk's pre-built authentication components.
- Manage user sessions without storing data in Sanity CMS.

Product Browsing:

- Fetch and display products from Sanity CMS using GROQ queries.

Cart Management:

- Use Context to manage cart state globally.
- Add/remove items and calculate totals dynamically.

Checkout Process:

- Collect user details and payment via Stripe-hosted checkout.
- Display order confirmation after successful payment.

Order Tracking:

- Generate a shipping label ID using ShipEngine.

Provide label ID to users for tracking.

3. API Endpoints:

Endpoint	Method	Purpose	Response Example
`/api/products`	Get	Fetch product data from Sanity CMS	`{ "id": 1, "name": "Product A", "price": 100, "stock": 20, "image": "url_to_image" }`
`/api/shipping-label`	Post	Generate a shipping label using ShipEngine	`{ "label ID": "LABEL123", "status": "Generated" }`
`/api/track-order`	Get	Retrieve order status using ShipEngine label ID	`{ "shipment ID": "SHIP123", "order ID": 123, "status": "In Transit", "expectedDeliveryDate": "2025-01-16" }`
`/api/checkout`	Post	Integrate Stripe for payment processing	`{ "order ID": 123, "status": "Success" }`

---

4. Sanity Schema Example:

```
export default {
  name: 'product',
  type: 'document',
  fields: [
    {name: 'name', type: 'string', title: 'Product Name'},
    {name: 'price', type: 'number', title: 'Price'},
    {name: 'stock', type: 'number', title: 'Stock Level'},
    {name: 'description', type: 'text', title: 'Description'},
  ],
};
```

- ## Conclusion:

  This documentation outlines the technical foundation of the Hecto e-commerce platform. It includes API endpoints, examples of API responses, and Sanity schema examples. This comprehensive overview will guide the implementation and development of the project effectively.