Name: Hammad Noor Khan

Roll No: 00457174

Day 2 – Technical Planning for [Hekto-Commerce Marketplace]

Define Technical Requirements

- **Frontend Requirements**:
  To create a seamless user experience, the interface will be simple, intuitive, and responsive for all devices. The following pages will be developed:
  - **Home**: A welcoming landing page showcasing featured products.
  - **Product Listing**: A page displaying all products with filtering and sorting options.
  - **Product Details**: A detailed page for each product with descriptions, images, and reviews.
  - **Cart**: A page summarizing selected items.
  - **Checkout**: A secure page for completing the purchase.
  - **Order Confirmation**: A final page confirming the order. ○ **Sanity CMS as Backend**:

  Sanity CMS will handle all backend operations, including:
  - Managing product data, customer details, and order records.
  - Schemas will be designed to meet the platform's functional needs, ensuring smooth alignment with frontend requirements.
- **Third-Party APIs**:
  APIs will be integrated to enhance functionality:
  - **Shipment Tracking**: Real-time updates on order delivery.
  - **Payment Gateways**: Secure and diverse payment options.
  - Backend services will be carefully chosen to ensure compatibility with frontend features.

Summary

The technical planning ensures all features align with the business goals from Day 1. By focusing on user experience, robust backend management, and reliable API integrations, the project is set up for successful development.

2. Design System Architecture

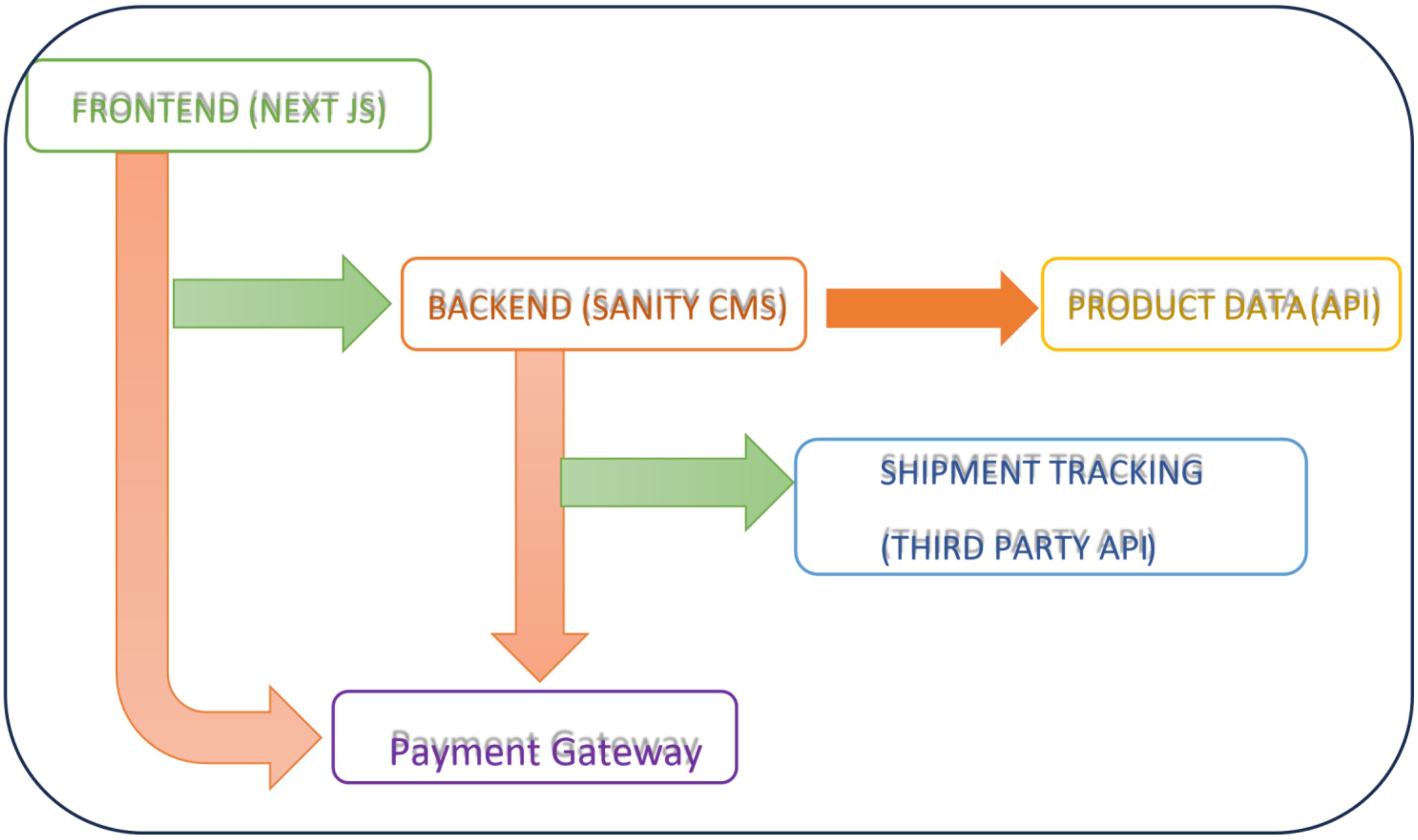
To ensure seamless data flow and efficient system interaction, the architecture will consist of the following components:

System Components and Workflow

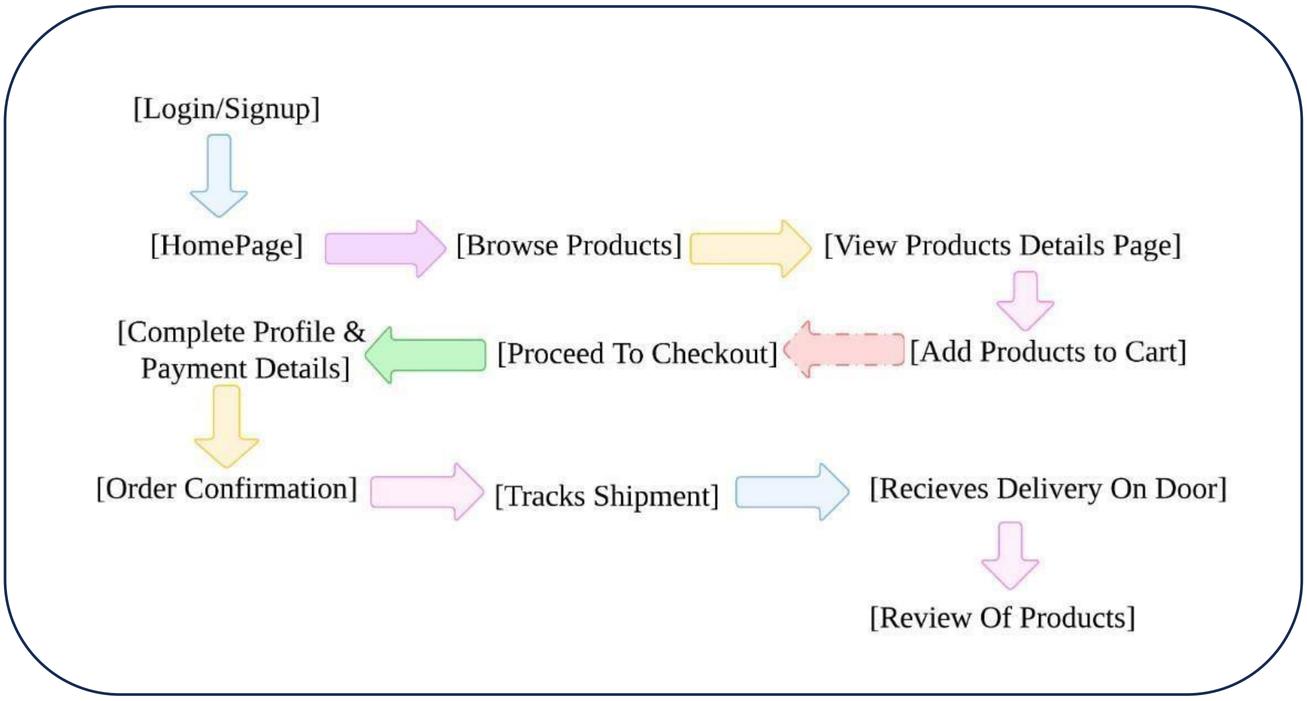
1. **Frontend (Next.js)**:
   - Provides the user interface for browsing products, placing orders, and tracking shipments.
   - Dynamically fetches data from the backend and renders it for users.
2. **Sanity CMS**:
   - Serves as the backend for managing product data, customer information, and order records.
   - Receives and stores order details via API requests from the frontend.
3. **Third-Party APIs**:
   - **Product Data API**:
     - ✦ Provides product listings and details from Sanity CMS to the frontend.
   - **Shipment Tracking API**:
     - ✦ Fetches real-time shipment updates and displays them to users.
   - **Payment Gateway**:
     - ✦ Processes payments securely and sends confirmation data to the frontend and Sanity CMS.

HIGH-LEVEL SYSTEM ARCHITECTURE DIAGRAM

This architecture ensures smooth interaction between the user interface, backend, and third-party services.



WORK FLOW DIAGRAM

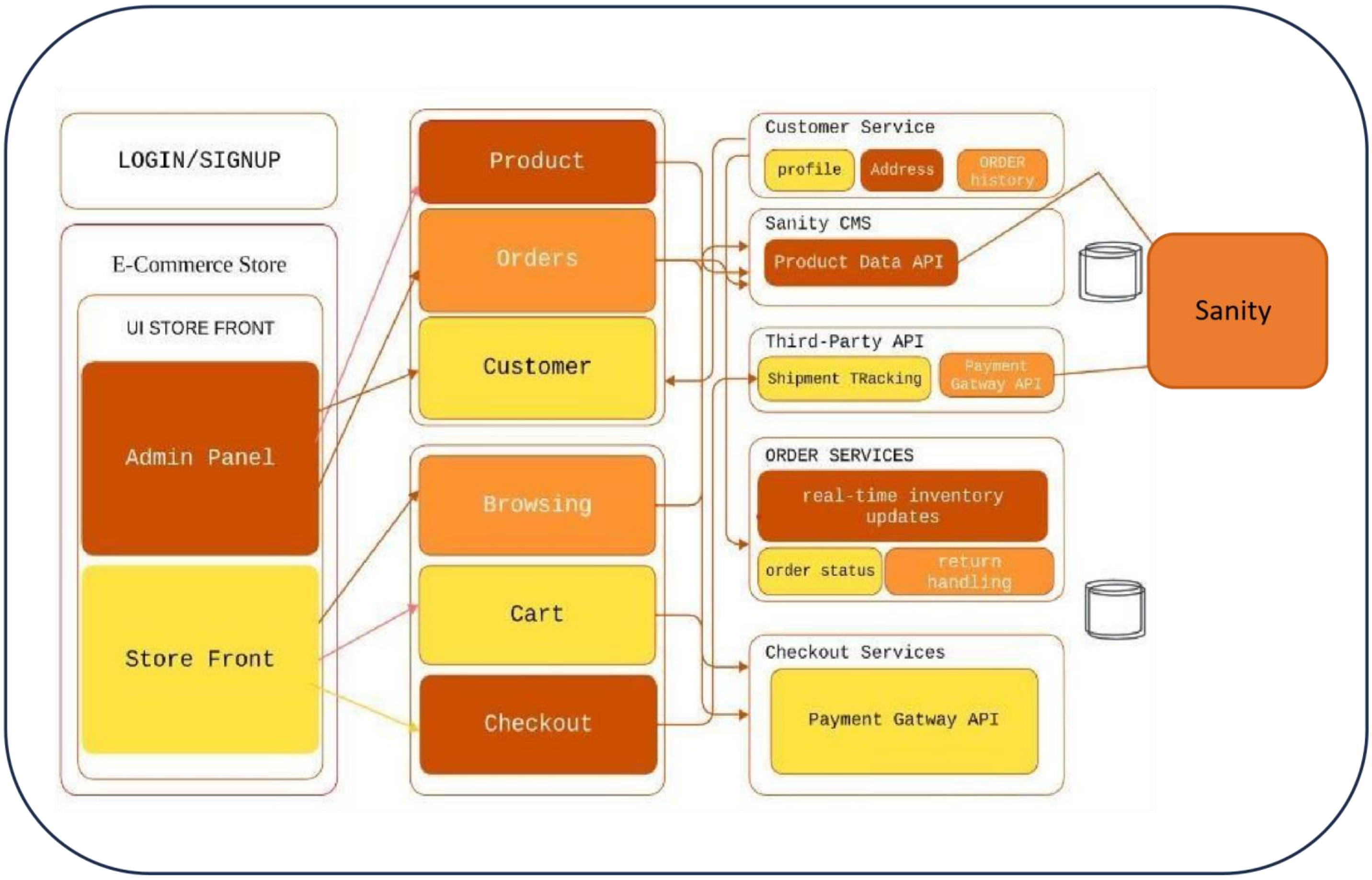


Workflow

1. **Browsing Products**:
    - A user visits the marketplace frontend.
    - The frontend sends a request to the Product Data API (powered by Sanity CMS).
    - Products are dynamically fetched and displayed on the website.
2. **Placing an Order**:
    - The user selects products and completes the order process.
    - Order details are sent to Sanity CMS via an API request for storage.
3. **Processing Payment**:
    - Payment details are securely processed by the payment gateway.
    - Confirmation is sent back to the frontend and recorded in Sanity CMS.
4. **Shipment Tracking**:

- Shipment information is fetched via a Third-Party API.
- The data is displayed on the user’s order tracking page in real-time.

SYSTEM ARCHITECTURE DIAGRAM



Detailed API Requirements:

This document outlines the API endpoints, methods, and expected responses required for the system to function efficiently, along with category-specific examples.

1. Product Data API

Base URL: https://api.example.com

Endpoint	Method	Description	Parameters	Response
/products	GET	Fetch all products.	category (optional)	List of all products.

/products/{id}	GET	Fetch details of a single product.	id (required)	Product details.
/categories	GET	Fetch all product categories.	None	List of categories.
/categories/{id}	GET	Fetch products in a specific category.	id (required)	Products under the category.

Example:

- **Request**:
  GET https://api.example.com/products?category=electronics
- **Response**:

```
[
 {
  "id": 1,
  "name": "Smartphone",
  "price": 299.99,
  "description": "A powerful smartphone with excellent features.",
  "category": "electronics"
 }
]
```

2. Order Management API

Endpoint	**Method**	**Description**	**Parameters**	**Response**
/orders	POST	Create a new order.	Order details (JSON)	Order confirmation.
/orders/{id}	GET	Fetch details of a specific order.	id (required)	Order details.
/orders	GET	Fetch all orders of a user.	user_id (optional)	List of user’s orders.

Example:

- **Request**:
  POST https://api.example.com/orders
- **Request Body**:

```
{
 "user_id": 123,
 "products": [
  { "product_id": 1, "quantity": 2 },
  { "product_id": 3, "quantity": 1 }
 ],
 "total_price": 99.97
}
```

- **Response**:

```
{
 "order_id": 456,
 "status": "Order placed successfully"
}
```

3. Shipment Tracking API

Endpoint	**Method**	**Description**	**Parameters**	**Response**
/shipment/{order_id}	GET	Fetch shipment status for an order.	order_id (required)	Shipment tracking details.

Example:

- **Request**:
  GET https://api.example.com/shipment/456
- **Response**:

```
{
 "order_id": 456,
 "status": "In Transit",
 "estimated_delivery": "2025-01-25"
}
```

4. Payment Gateway API

Endpoint	Method	Description	Parameters	Response
/payment	POST	Process payment.	Payment details (JSON)	Payment confirmation.

Example:

- **Request**:
  POST https://api.example.com/payment
- **Request Body**:

```
{
  "order_id": 456,
  "user_id": 123,
  "amount": 99.97,
  "payment_method": "credit_card"
}
```

- **Response**:

```
{
  "payment_id": 789,
  "status": "Payment Successful",
  "transaction_date": "2025-01-21"
}
```

Category-Specific Example

To fetch products under the "Electronics" category:

- **Request**:
  GET https://api.example.com/categories/electronics
- **Response**:

```
[
  {
    "id": 101,
    "name": "Laptop",
    "price": 999.99,
    "description": "High-performance laptop.",
    "category": "electronics"
  },
  {
    "id": 102,
    "name": "Headphones",
    "price": 49.99,
    "description": "Noise-cancelling headphones.",
    "category": "electronics"
  }
]
```

This document provides a detailed guide for understanding the API endpoints, ensuring effective integration with both the frontend and backend components of the system.

SCHEMA (DRAFTED)

```
import { defineType,
defineField } from "sanity"; export const
product = defineType({    name: "product",
title: "Product",  type:
"document",
fields: [
   {      name: "title",     title:
"Title",     validation: (rule) =>
rule.required(),     type: "string",
   },
   //slug field
defineField({
name: "slug",     type:
"slug",     title: "Slug",
options: {      source:
"title",
maxLength: 96,
     },
     validation: (Rule) => Rule.required(),
   }),
   {
     name: "description",      type: "text",
validation: (rule) => rule.required(),      title:
"Description",
   },
   {
     name: "productImage",      type:
"image",      validation:
(rule) => rule.required(),      title:
"Product Image",
   },
   {
     name: "price",      type: "number",
validation: (rule) => rule.required(),      title:
"Price",
   },
   {
     name: "tags",      type:
"array",      title:
"Tags",      of: [{ type:
"string" }],
   },
   {
     name: "dicountPercentage",
type: "number",      title:
"Discount Percentage",
   },
   {
     name: "isNew",      type:
"boolean",      title:
```

```
"New Badge",
   },
 ],
});
```

Achievements:

Successfully outlined the technical planning for the Bandage-E-Commerce Marketplace, including a responsive frontend (Next.js) with essential pages, a robust backend powered by Sanity CMS for seamless data management, and integration of APIs for shipment tracking, secure payments, and product data. Designed detailed workflows, API requirements, and schema drafts to ensure smooth system functionality and scalability.