HACKATHON 3

Day 2

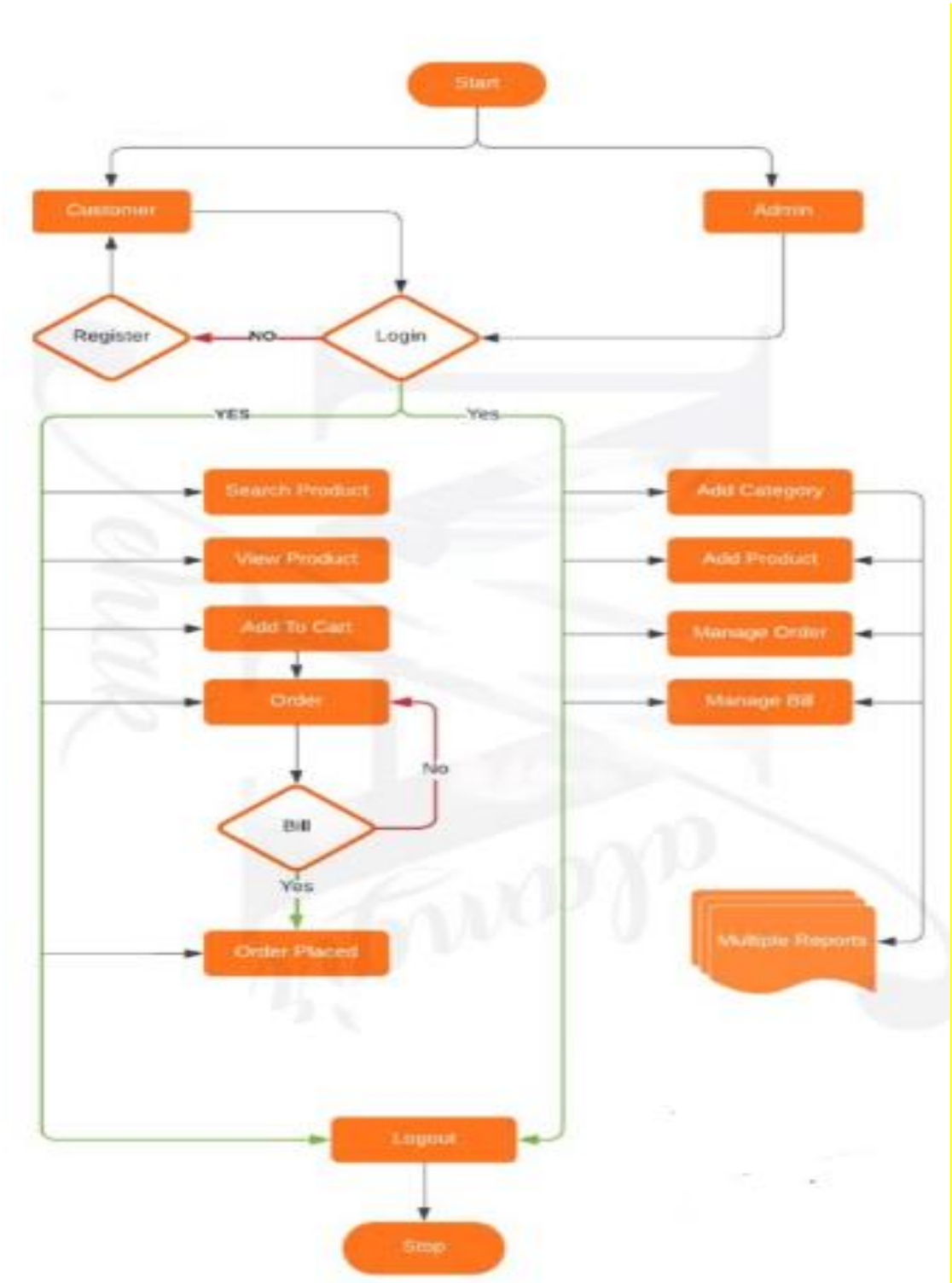
Frontend Requirements:

A responsive, user-friendly interface for browsing dairy and bakery products.

Pages for Home, Product Listing, Product Details, Cart, Checkout, and Order Confirmation.

Ensure smooth integration with the backend for product data display and checkout functionality.

Flow chart


Start
Customer
Admin
Register
NO
Login
YES
Yes
Search Product
View Product
Add To Cart
Order
No
Bill
Yes
Order Placed
Add Category
Add Product
Manage Order
Manage Bill
Multiple Reports
Logout
Stop

Sanity CMS as Backend:

Use Sanity CMS to manage product information, customer details, and order records.

Design CMS schemas that support the quick-commerce nature of the marketplace, ensuring easy management of perishable product data.

Create reference relationships in Sanity to track product categories, brands, and availability.

Third-Party APIs:

Integrate external APIs for payment processing, shipment tracking, and real-time inventory updates.

Ensure APIs support the necessary data flow from frontend to backend and third-party services.

2. Design System Architecture

Create a high-level diagram of the system architecture that includes the following components:

Frontend (Next.js): **The user interface where customers interact with the platform.**

Sanity CMS: **Backend CMS for managing product data and customer details.**

Third-Party APIs**: External APIs used for payment processing and shipment tracking.**

Example System Flow:

A customer browses the website and selects products.

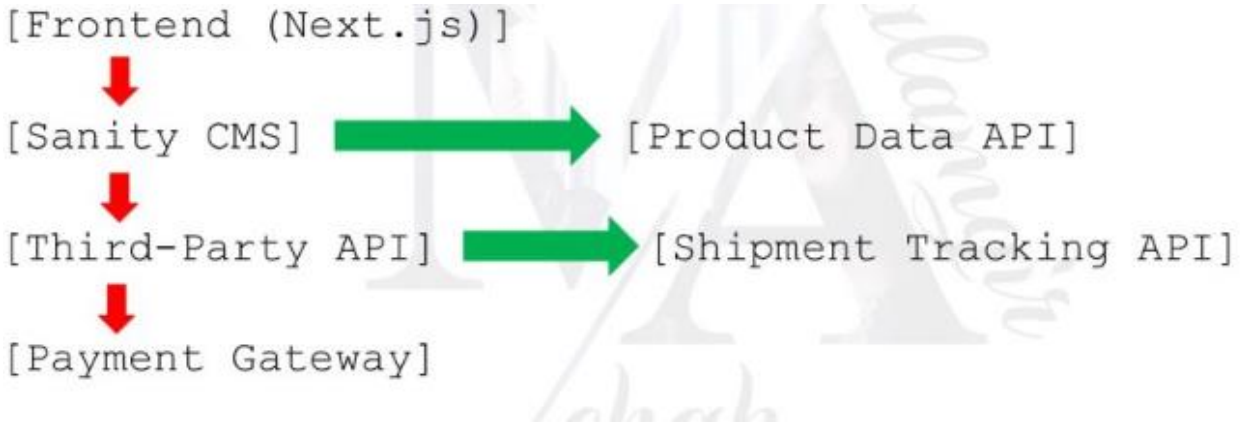
The frontend communicates with the Sanity CMS to fetch product data, including pricing, stock, and descriptions.

When the user adds products to the cart and proceeds to checkout, order details are saved in the Sanity CMS via API.

Shipment tracking is integrated using a third-party API that provides real-time updates on order status.

Payment information is securely processed via a payment gateway, and the transaction is recorded in Sanity CMS.

System Diagram Example:



3. Plan API Requirements

Define API endpoints based on the product schema and business requirements.

Endpoint Name: /products

Method: **GET**

Description: **Fetch all available products.**

Response Example: { "id": 1, "name": "Fresh Milk", "price": 150, "stock": 30, "image": "link_to_image" }

Endpoint Name: /orders

Method: **POST**

Description: **Create a new order.**

Payload: { "customerId": 123, "products": [{ "productId": 1, "quantity": 2 }], "paymentStatus": "pending" }

Response Example: { "orderId": 456, "status": "created" }

Endpoint Name: /shipment

Method: **GET**

Description: **Fetch shipment tracking details.**

Response Example: { "orderId": 456, "status": "In Transit", "ETA": "30 minutes" }

4. Write Technical Documentation

Document all technical decisions, including system architecture, API requirements, and CMS schema designs.

System Architecture Overview:

Include a clear diagram of how the frontend interacts with Sanity CMS and third-party APIs, providing a high-level view of data flow and interactions.

Key Workflows:

User Registration:

The user signs up, and data is stored in Sanity CMS, with a confirmation sent to the user.

Product Browsing:

The user views product categories, and the frontend fetches product details from Sanity CMS.

Order Placement:

The user adds products to the cart and proceeds to checkout. Order details are saved in Sanity CMS.

Shipment Tracking:

The user views live tracking updates from the third-party API.

API Documentation:

List all API endpoints, their methods, and response examples, ensuring clear communication on how each part of the system communicates.

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: 'image', type: 'image', title: 'Product Image' }
  ]
};
```

DIAGRAM