HACKATHON 3
DAY-2

Marketplace Technical Foundation

On Day 2 of Hackathon 3, the marketplace's technical foundation was established with Next.js for the frontend, Sanity CMS for content management, and integrations like a Payment Gateway and Shipment Tracking API. The setup focused on modularity and scalability, enabling smooth product data handling, secure transactions, and real-time order tracking.

NAME: NIMRA RAZI

ROLL NO.: 0045228

QUARTER 2

1/2/2025

Title:

“Marketplace Technical Foundation-[Style Haven E-Commerce].”

User Journey in the Marketplace:

1. User Visits the Marketplace (Homepage)

- The user lands on the homepage built with **Next.js**, which fetches dynamic content from **Sanity CMS**.
- They browse featured products, promotions, and categories.

2. Product Discovery & Listing Page

- The user navigates to a category or searches for products.
- The frontend calls the **Sanity CMS API** to fetch filtered products.
- Sorting, filtering, and pagination features are available for a seamless browsing experience.

3. Product Detail Page (PDP)

- Clicking on a product takes the user to its detailed view.
- Product images, descriptions, reviews, and availability are dynamically loaded.
- The frontend makes API calls to retrieve real-time stock information and estimated delivery times from **third-party APIs** (e.g., a shipping provider).

4. Add to Cart & Cart Management

- The user adds items to the cart, which is managed via a **state management system (React Context/Redux/Zustand)**.
- The cart updates dynamically and is stored in **local storage/session storage** to persist data across sessions.
- Users can modify item quantities or remove products before proceeding to checkout.

5. Checkout Process

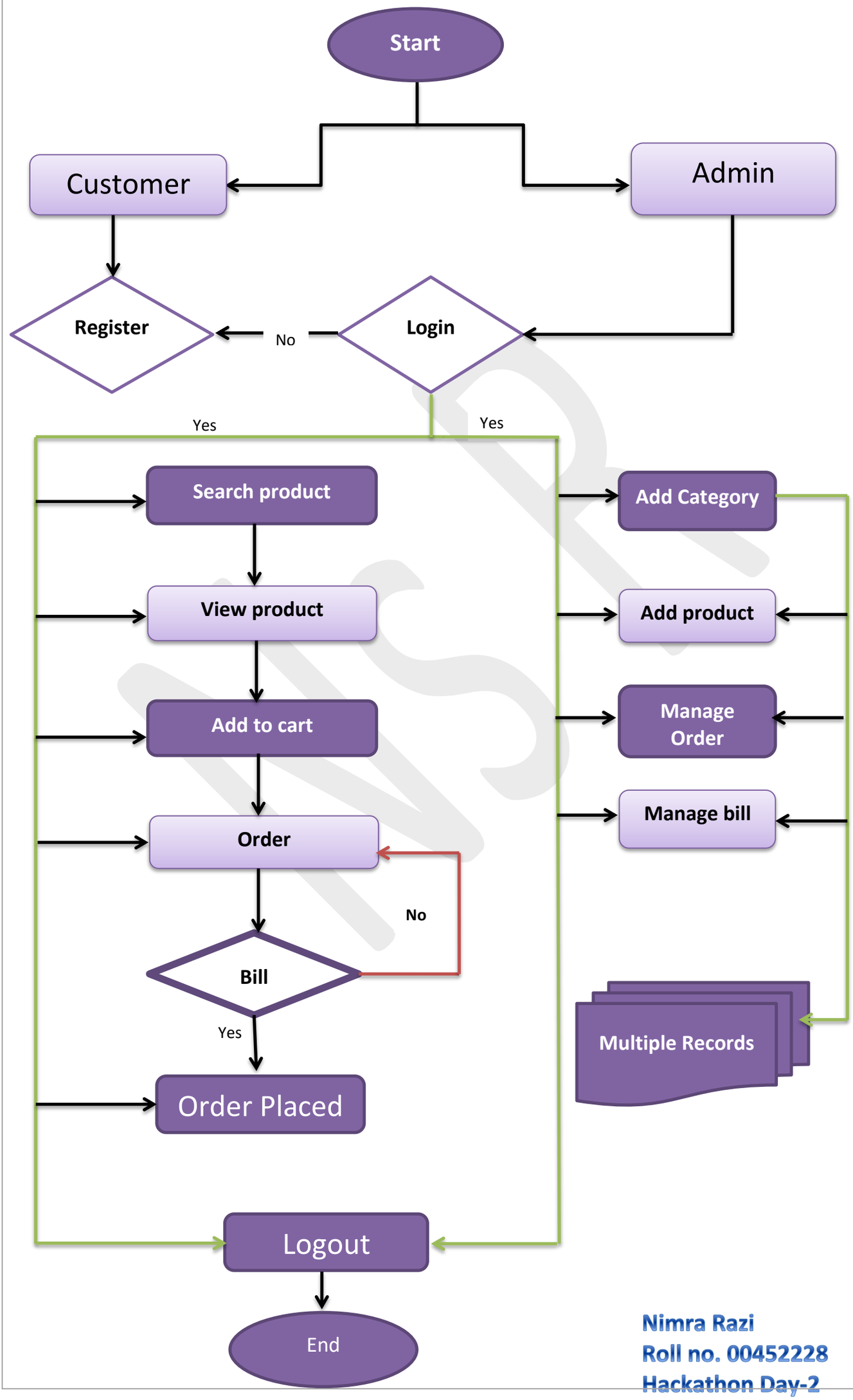
- The user enters shipping details and selects a preferred payment method.
- The frontend validates the form and sends order data to the backend (**Sanity CMS order schema**).
- Payment gateway integration (e.g., **Stripe, PayPal**) handles the transaction.
- Once payment is successful, an order confirmation is generated.

6. Order Confirmation & Tracking

- After successful payment, the order is recorded in **Sanity CMS**.
- The user receives an email confirmation via a third-party service (e.g., **Send Grid, Mail chimp**).
- The order status updates dynamically using shipment tracking APIs.
- Users can visit the "My Orders" page to check their order history and track deliveries.

Step	Frontend (Next.js, Tailwind	Backend (Sanity CMS, APIs)	Third-Party APIs
Homepage & Product Listing	Responsive UI, dynamic content	Fetch categories/products from Sanity	-
Product Detail Page	Fetch product details dynamically	Store product data, manage availability	Inventory API for stock updates
Cart & Checkout	State management, local storage	Create order schema, validate user data	Payment gateway API (Stripe, PayPal)
Order Confirmation	Display confirmation UI	Store order info in Sanity CMS	Email API for confirmation
Order Tracking	Fetch order status dynamically	Store tracking ID, fetch status updates	Shipment tracking API

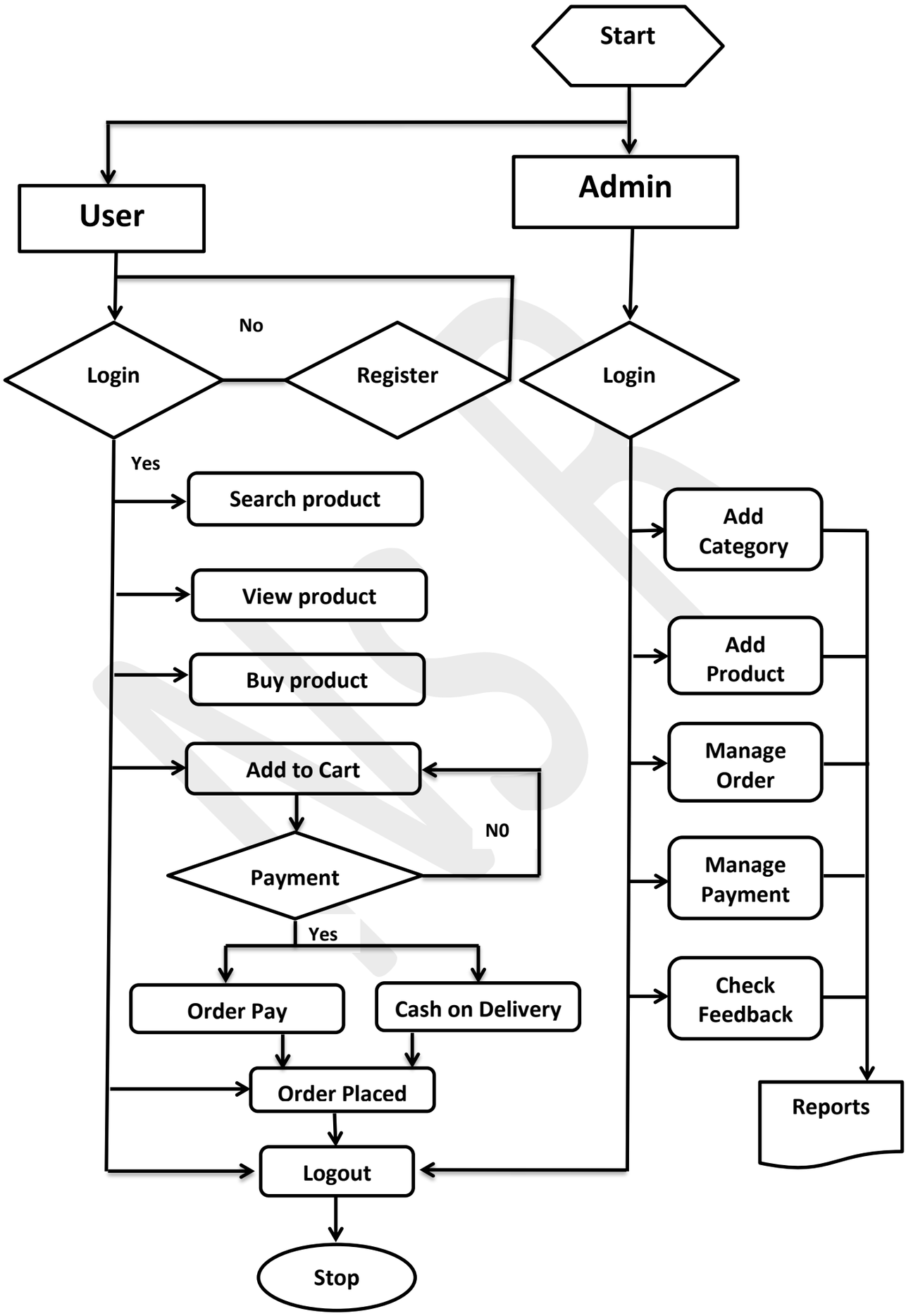
Frontend Requirements: Flow Chart of User



Nimra Razi
Roll no. 00452228
Hackathon Day-2

Frontend Requirements:

Flow Chart of User



Nimra Razi
Roll no. 00452228
Hackathon Day-2

Frontend Requirements:

1. **Core Pages & UI** – Responsive Next.js UI with Tailwind CSS for **Home, Product Listing, Product Details, Cart, Checkout, and Order Confirmation**.
2. **State Management & Performance** – Use **React Context API, Zustand, or Redux** for cart, auth, and orders with local storage persistence.
3. **API Integrations** – Fetch dynamic content from **Sanity CMS** and integrate **third-party APIs** for payments, shipping, and order tracking.
4. **Form Handling & Validation** – Secure user inputs using **React Hook Form + Zod** for checkout and authentication.
5. **Animations & UX** – Implement **Framer Motion** for smooth UI interactions and enhance user **experience.**

System Architecture:

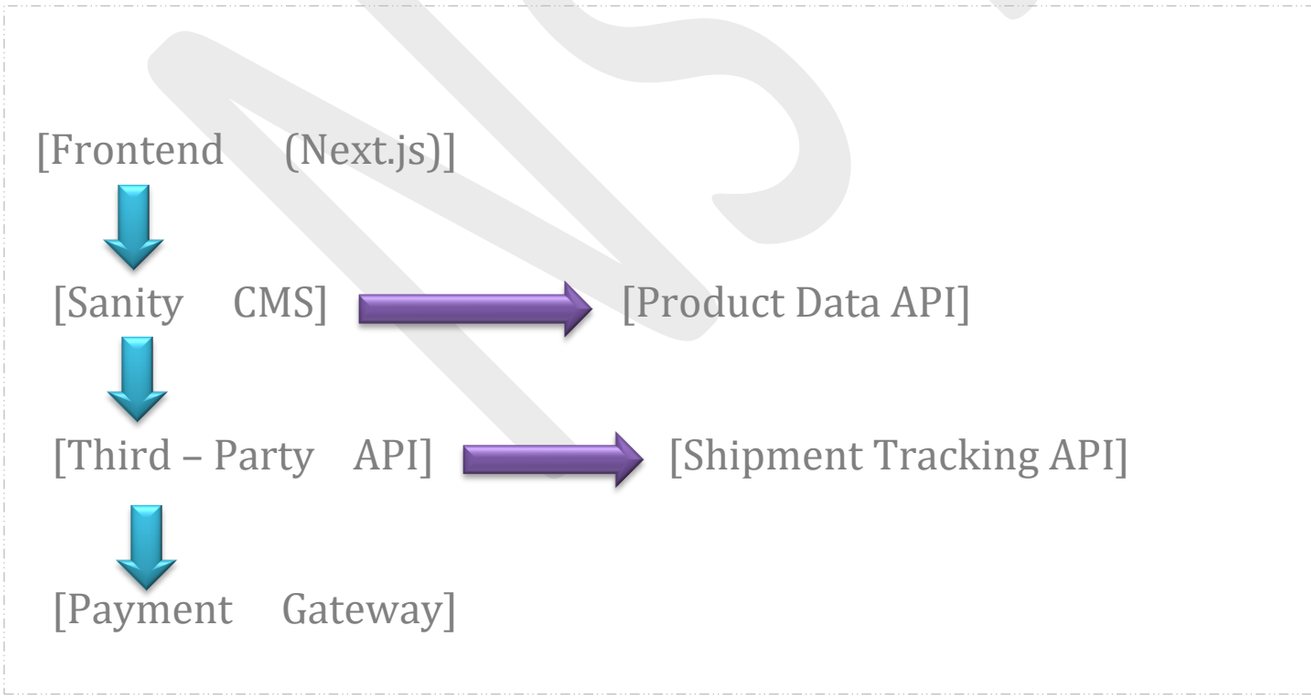
System Architecture to visualize how your marketplace components interact:

Components & Data Flow

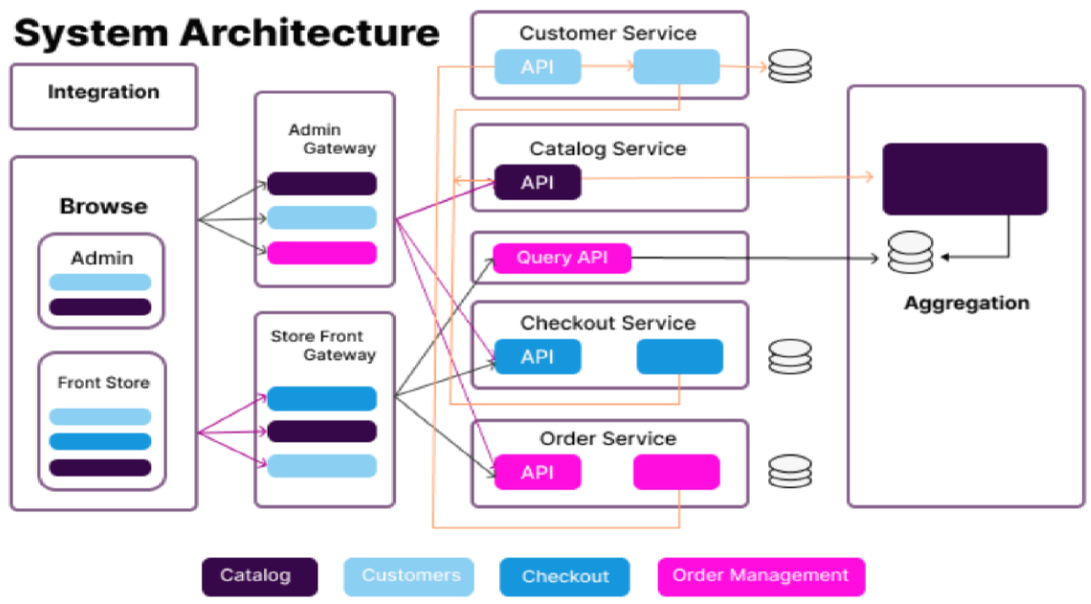
1. **User interacts with the frontend** (Next.js).
2. **Frontend fetches data from Sanity CMS** (Product listings, user data, and orders).
3. **Orders are stored in Sanity CMS.**
4. **Shipment tracking is fetched via a Third-Party API.**
5. **Payments are processed through a Payment Gateway.**

System Architecture Overview

Diagram:

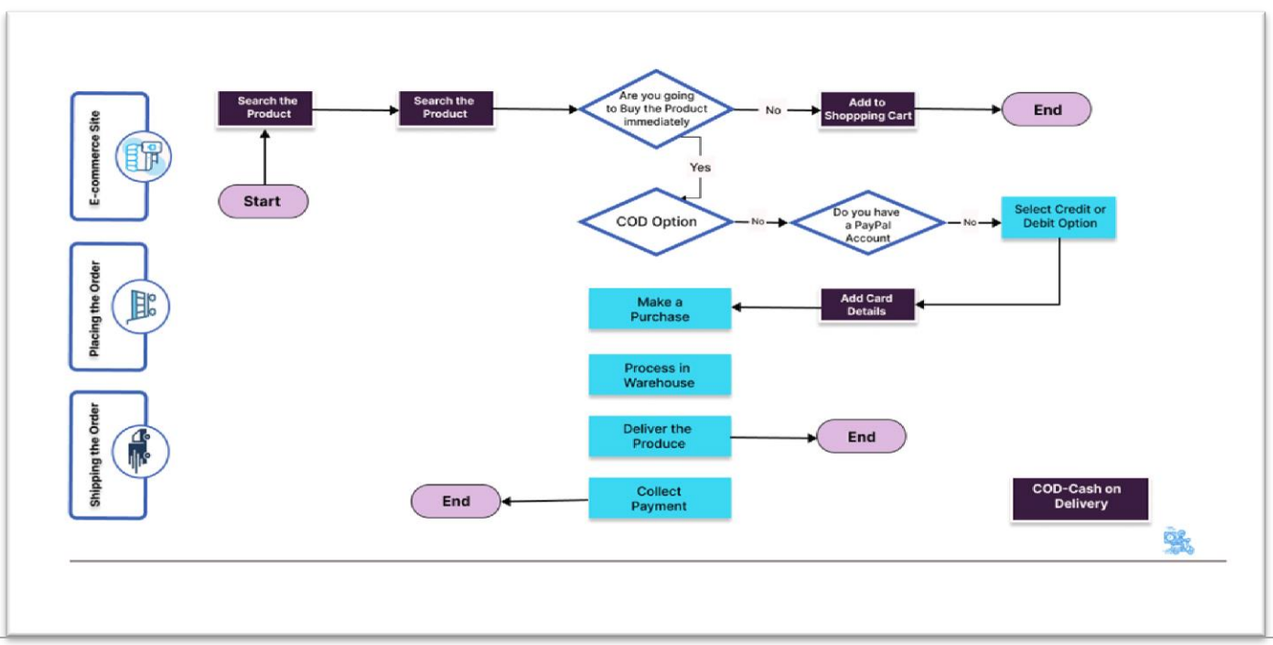


System Architecture Diagram:



Components Roles:

1. ***Frontend (Next.js)***
   - Builds a responsive UI for pages like Home, Product Listing, Cart, and Checkout.
   - Fetches product and user data via APIs.
   - Handles user interactions, order placement, and navigation.
2. ***Sanity CMS***
   - Manages product, user, and order data.
   - Provides structured content via the Product Data API.
   - Ensures scalability and easy content updates.
3. ***Product Data API***
   - Fetches product details (name, price, images, stock, etc.).
   - Enables dynamic content rendering on the frontend.
4. ***Third-Party API***
   - Integrates shipment tracking for order status updates.
   - Provides real-time logistics data for users.
5. ***Payment Gateway***
   - Processes secure online payments.
   - Handles transactions, refunds, and payment verification.



ER Diagram and Entities:

This diagram will define the relationships between entities in your database.

Example Entities:

1. Furniture

Fields: id, name, description, price, categoryId, stock, material, imageUrls, rating.

2. Category

Fields: id, name, description, parentId.

3. User

Fields: id, name, email, password, address, phone, role.

4. Order

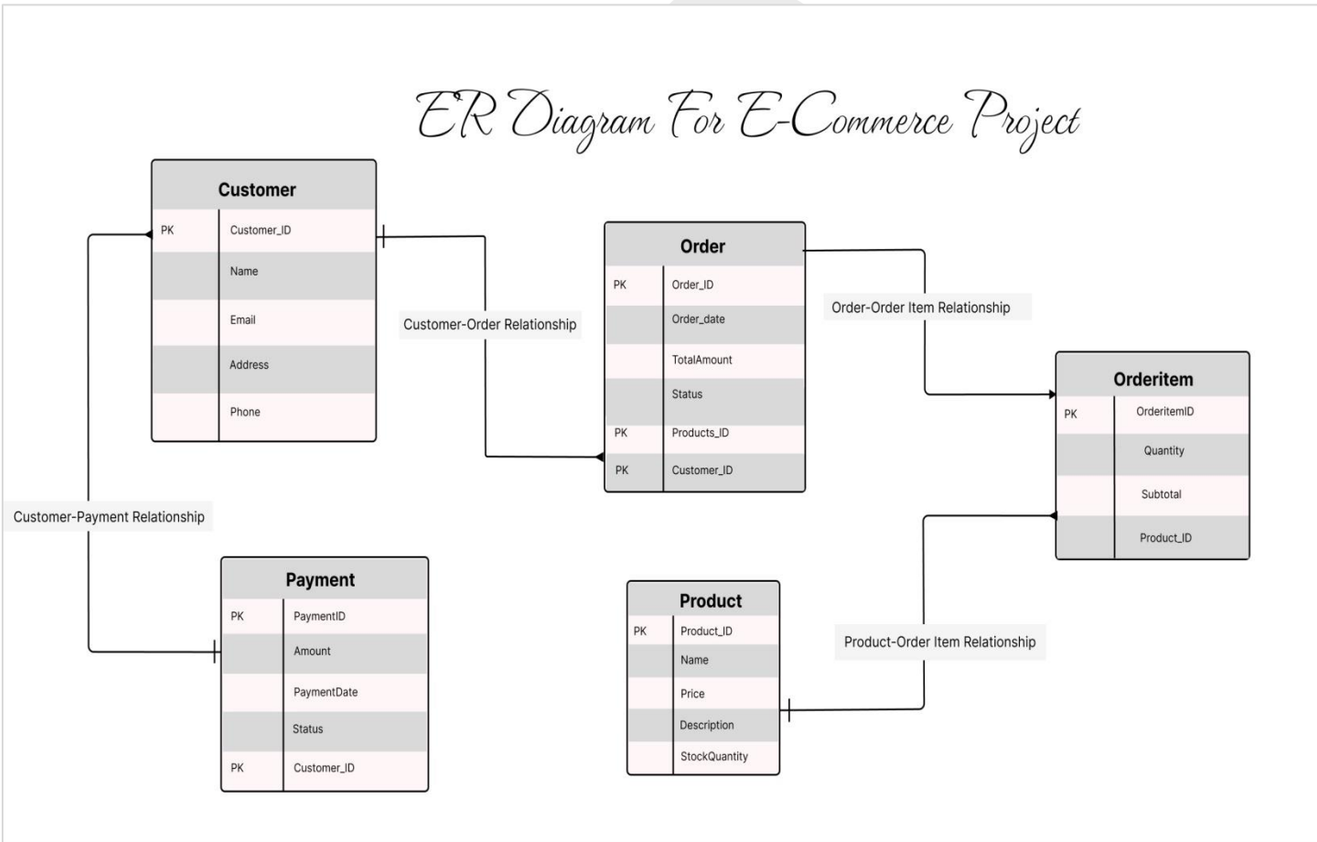
Fields: id, userId, orderDate, status, totalAmount.

5. OrderItem

Fields: id, orderId, furnitureId, quantity, price.

6. Review

Fields: id, userId, furnitureId, rating, comment.

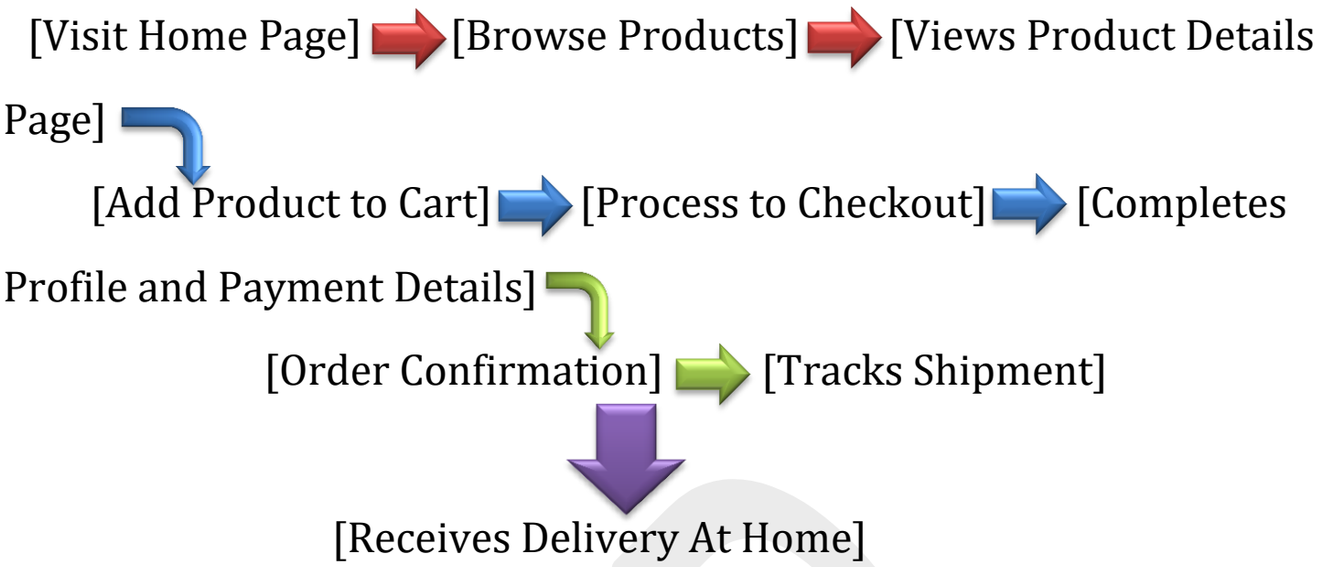


API ENDPOINTS:

API EndPoints.xls

Endpoint	Method	Description	Parameters	Response Example
/api/furniture	GET	Fetch all furniture items	None	{ id: 1, name: "Sofa" }
/api/furniture/:id	GET	Fetch a single furniture item	id (Path)	{ id: 1, name: "Sofa" }
/api/furniture	POST	Add a new furniture item	name, price, category(Body)	{ success: true, id: 5 }
/api/furniture/:id	PUT	Update a furniture item	id (Path), name, price(Body)	{ success: true }
/api/furniture/:id	DELETE	Delete a furniture item	id (Path)	{ success: true }
/api/categories	GET	Fetch all furniture categories	None	{ categories: ["Living Room"] }

Work Flow Diagram Visual Representation



Sanity Schema.js:

```
export default createSchema({
  name: 'default',
  types: schemaTypes.concat([
    // Furniture Product Schema
    {
      name: 'product',
      title: 'Product',
      type: 'document',
      fields: [
        { name: 'name', title: 'Name', type: 'string' },
        { name: 'slug', title: 'Slug', type: 'slug', options: { source: 'name', maxLength: 96 } },
        { name: 'price', title: 'Price', type: 'number' },
        { name: 'description', title: 'Description', type: 'text' },
        { name: 'category', title: 'Category', type: 'string' },
        { name: 'dimensions', title: 'Dimensions', type: 'string' },
        { name: 'material', title: 'Material', type: 'string' },
        { name: 'image', title: 'Image', type: 'image', options: { hotspot: true } },
        { name: 'stock', title: 'Stock', type: 'number' },
        { name: 'tags', title: 'Tags', type: 'array', of: [{ type: 'string' }] },
      ],
    },
    // Order Schema
    {
      name: 'order',
      title: 'Order',
      type: 'document',
      fields: [
        { name: 'customer', title: 'Customer', type: 'reference', to: [{ type: 'customer' }] },
        { name: 'orderDate', title: 'Order Date', type: 'datetime' },
        { name: 'status', title: 'Status', type: 'string', options: { list: ['Pending', 'Processing', 'Shipped', 'Delivered'] } },
        { name: 'total', title: 'Total', type: 'number' },
        {
          name: 'items',
          title: 'Items',
          type: 'array',
          of: [
            {
              type: 'object',
              fields: [
                { name: 'product', title: 'Product', type: 'reference', to: [{ type: 'product' }] },
                { name: 'quantity', title: 'Quantity', type: 'number' },
```

```
            ],
          },
        ],
      },
    ],
  },
  // Customer Schema
  {
    name: 'customer',
    title: 'Customer',
    type: 'document',
    fields: [
      { name: 'name', title: 'Name', type: 'string' },
      { name: 'email', title: 'Email', type: 'string' },
      { name: 'phone', title: 'Phone', type: 'string' },
      { name: 'address', title: 'Address', type: 'object', fields: [
          { name: 'street', title: 'Street', type: 'string' },
          { name: 'city', title: 'City', type: 'string' },
          { name: 'state', title: 'State', type: 'string' },
          { name: 'postalCode', title: 'Postal Code', type: 'string' },
        ],
      },
    ],
  },
 ]),
});
```

THE END