BOOK STORE GIAIC MARKET READY HACKTHON 3 – DAY 3 SAKEENA FATIMA 00466831

[Company name] | [Company address]

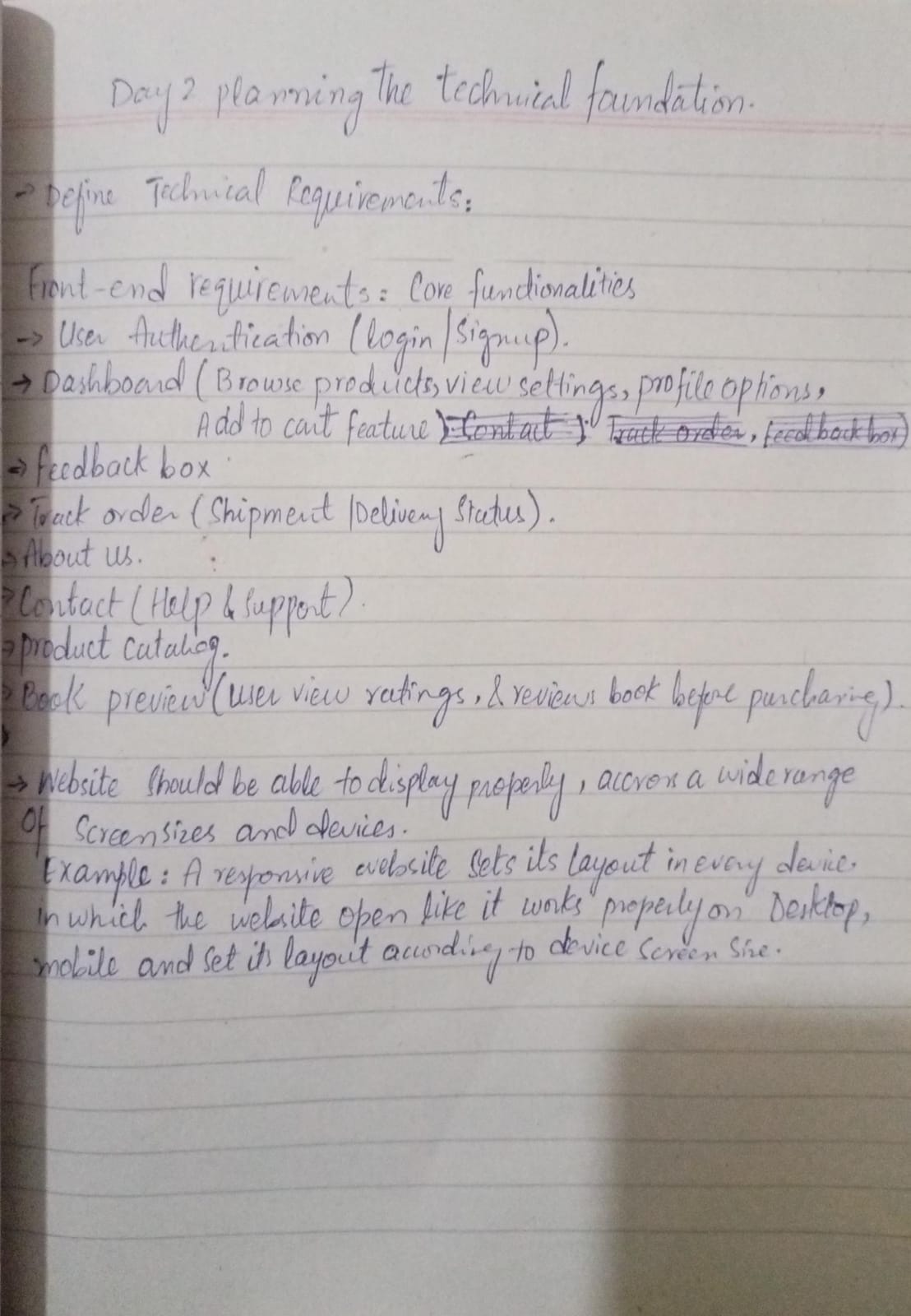
SAKEENA FATIMA - 00466831

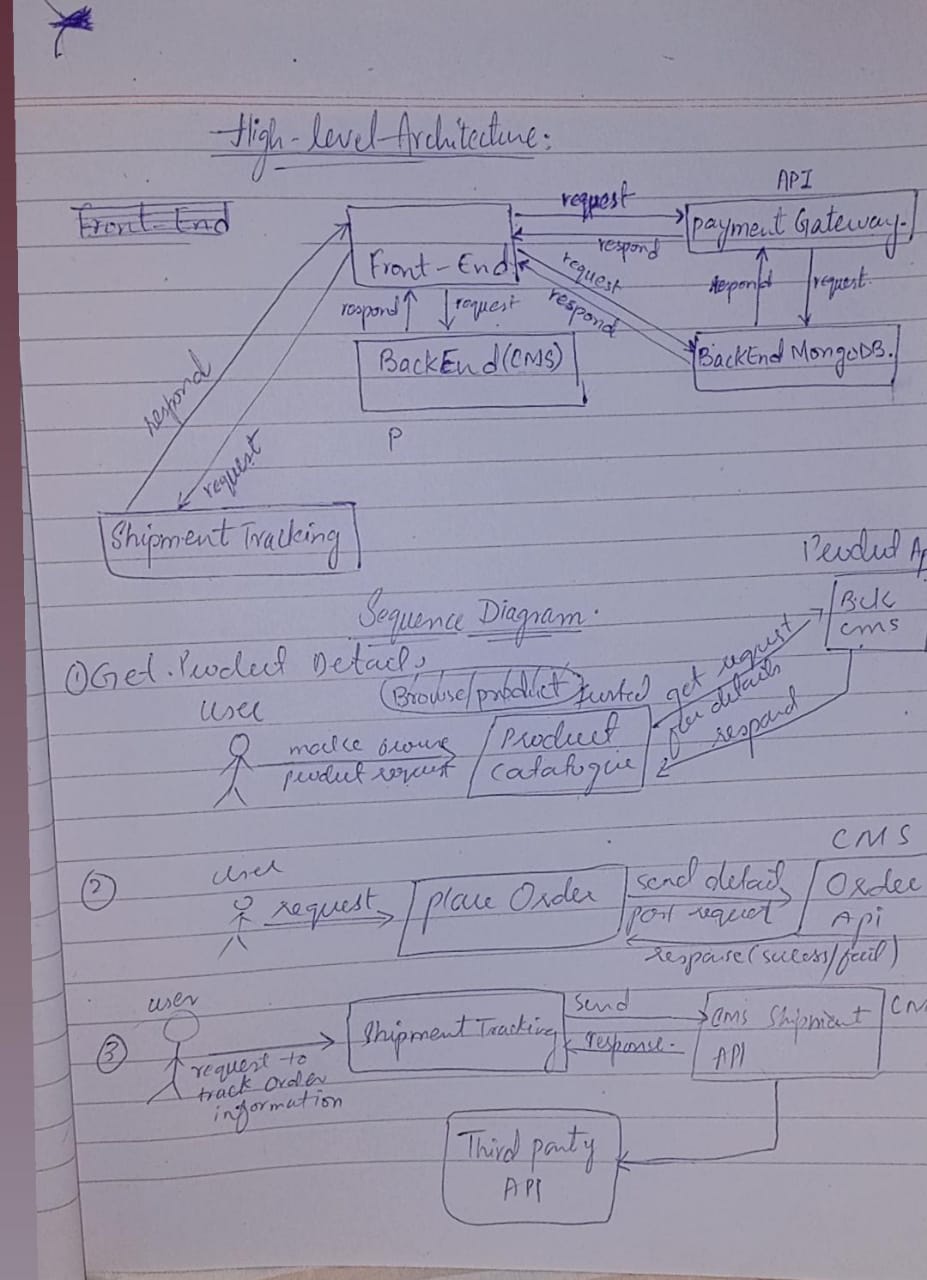
MARKET PLACE E-COMMERCE

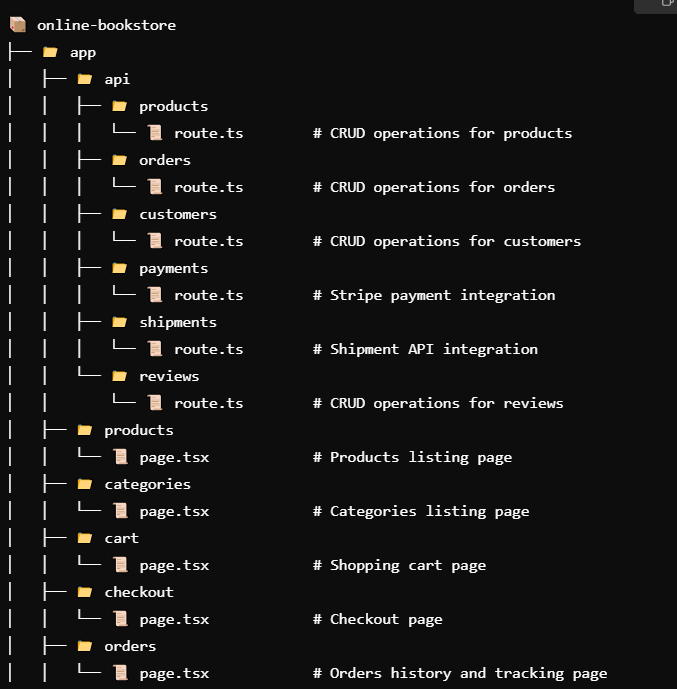
2025

**Day 2 Activities: Transitioning to Technical Planning**

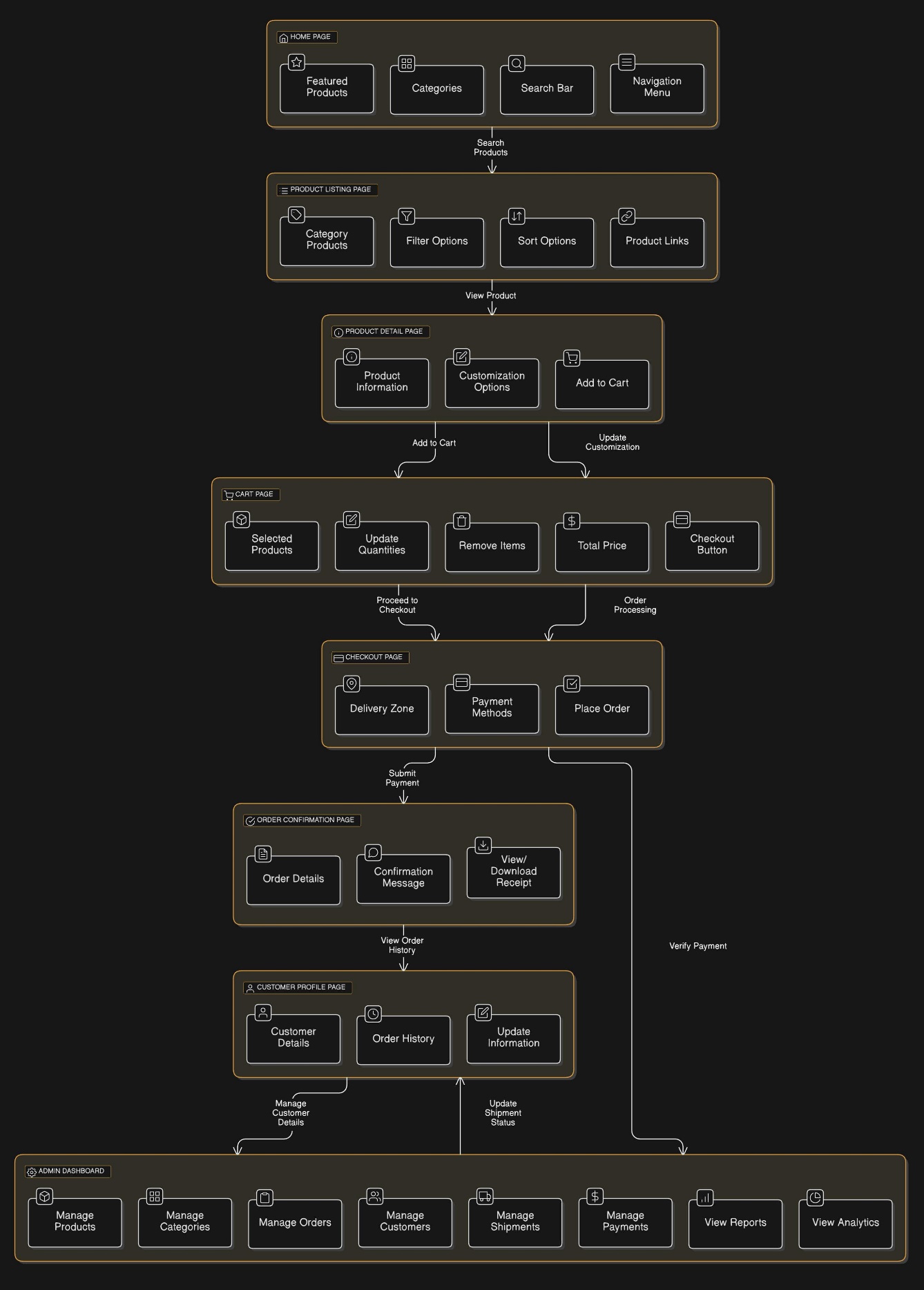
1. **Define Technical Requirements**

****



**2. Folder Structure**

**3. Design System Architecture**

****

### **Explanation: Detailed Component Interaction**

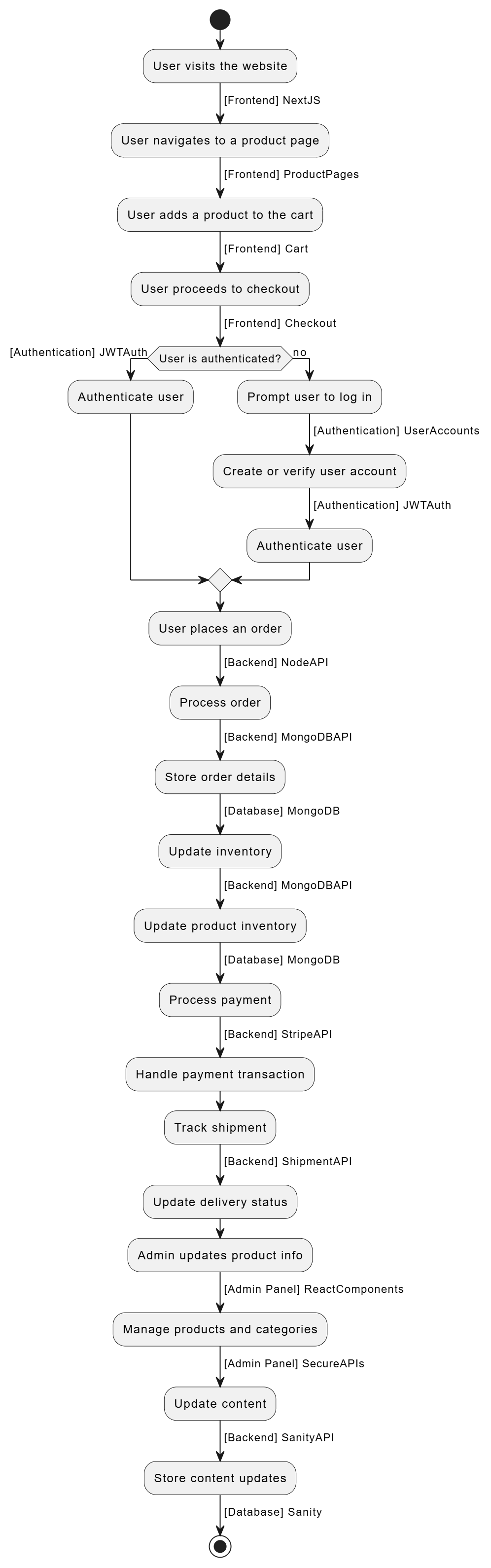
#### ****Purpose:****

This diagram dives deeper into the user journey, mapping the specific **UI components** and **backend integrations** across various pages.

#### ****Pages and Features:****

1. **Home Page:**
   * Contains features like:
     + **Featured Products**
     + **Categories**
     + **Search Bar**
2. **Product Listing Page:**
   * Users can view products filtered or sorted by categories.
   * **Backend Integration:** Queries to fetch filtered data based on user preferences.
3. **Product Detail Page:**
   * Displays:
     + **Product Information**
     + **Customization Options**
     + **Add to Cart Button**
4. **Cart Page:**
   * Features:
     + Update or remove items.
     + Calculate total price dynamically.
5. **Checkout Page:**
   * Handles:
     + **Delivery Zone Selection**
     + **Payment Methods**
     + **Order Submission**
   * **Payment Gateway (StripeAPI):**
     + SRE ensures payment API failures are logged with retries and fallback mechanisms.
6. **Order Confirmation Page:**
   * Displays:
     + **Order Details**
     + Confirmation messages and receipts.
7. **Customer Profile Page:**
   * Includes:
     + **Order History**
     + Options to update personal details.
8. **Admin Dashboard:**
   * Features:
     + Product/Category/Order management.
     + Payment and shipment tracking.
     + Reports and analytics.

**4. Work Flow Diagram**

****

### **Explanation: High-Level Workflow (Flowchart)**

#### ****Purpose:****

This flowchart outlines the user journey through the bookstore system, from visiting the website to placing an order and handling backend processes like inventory updates, payments, and content management.

#### ****Detailed Steps:****

1. **User Visits the Website:**
   * **Frontend Framework:** The website is built using **Next.js**, optimized for fast performance and scalability.
2. **User Navigates to a Product Page:**
   * Page rendering happens through **ProductPages** components.
3. **User Adds a Product to the Cart:**
   * The **Cart** component stores the selected product information.
4. **User Proceeds to Checkout:**
   * The **Checkout** module handles gathering shipping, billing, and payment details.
   * Security measures such as HTTPS encryption, monitoring fraud detection systems, and API rate limiting are crucial.
5. **Authentication Check:**
   * **JWTAuth** verifies the user's identity.
     + If not authenticated:
       - The user is prompted to **log in or create an account** via the **UserAccounts** module.
       - JWT tokens are issued upon successful authentication.
     + If authenticated:
       - Proceeds to the order process.
6. **Order Processing:**
   * The **Node API** manages requests to backend services like:
     + **MongoDBAPI:** Handles database transactions, such as saving order details and updating inventory.
7. **Inventory Update:**
   * Product inventory is updated in **MongoDB** after every successful order.
   * Regular backups and replication are essential to avoid data loss.
8. **Payment Processing:**
   * Payments are managed using **StripeAPI**.
9. **Shipment Tracking:**
   * After payment success, the shipment is tracked via **ShipmentAPI**.
10. **Admin Functionality:**
    * Admins update product details, manage categories, and store content using:
      + **SanityAPI** for CMS (content management system).
      + **SecureAPIs** for admin actions.

**5. Plan API Requirements**

**1. Product Management**

1. **Get All Products**
   * **Endpoint:** /api/products
   * **Method:** GET
   * **Purpose:** Retrieve the list of all available books stored in Sanity.
2. **Get Product by ID**
   * **Endpoint:** /api/products/:id
   * **Method:** GET
   * **Purpose:** Fetch specific book details using its unique identifier.

**Update Product Details (Admin)**

* + **Endpoint:** /api/products/:id
  + **Method:** PUT
  + **Purpose:** Allow admins to edit book information, such as price, stock, or title, in Sanity.
  + **Request Body Example:**

**2. Order Management**

1. **Place a New Order**
   * **Endpoint:** /api/orders
   * **Method:** POST
   * **Purpose:** Create a new order in the database with customer details, selected books, and payment status.
2. **View Order Details**
   * **Endpoint:** /api/orders/:id
   * **Method:** GET
   * **Purpose:** Retrieve the details of an order using its ID.

**3. Shipment Management**

1. **Track Shipment**
   * **Endpoint:** /api/shipments/:id
   * **Method:** GET
   * **Purpose:** Check the status of an order’s shipment using a third-party API.
2. **Update Shipment Status**
   * **Endpoint:** /api/shipments/:id
   * **Method:** PATCH
   * **Purpose:** Update the shipment status via webhook or admin input

**4. Payment Management**

1. **Process Payment**
   * **Endpoint:** /api/payments
   * **Method:** POST
   * **Purpose:** Handle payment transactions securely.
2. **Check Payment Status**
   * **Endpoint:** /api/payments/:id
   * **Method:** GET
   * **Purpose:** Verify the payment status for an order.

**5. Review Management**

1. **Submit a Book Review**
   * **Endpoint:** /api/reviews
   * **Method:** POST
   * **Purpose:** Allow customers to leave reviews for purchased books.
2. **Fetch Reviews for a Book**
   * **Endpoint:** /api/reviews/:bookId
   * **Method:** GET
   * **Purpose:** Display all reviews for a specific book.

# Marketplace Technical Foundation - Bookstore Marketplace

## 1. System Architecture Overview

### Component Descriptions

1. **Frontend (React/Next.js)**
   * Provides the user interface for browsing, searching, and purchasing books.
   * Interacts with the backend via API calls.
   * Manages state and user sessions.
2. **API Gateway**
   * Centralized entry point for all API requests.
   * Handles routing to appropriate backend services.
3. **Backend (Node.js/Express)**
   * Processes requests from the frontend.
   * Manages business logic such as order processing, user authentication, and data validation.
4. **Sanity CMS**
   * Used for managing book inventory, categories, and author details.
   * Provides APIs for dynamic content management.
5. **Database (MongoDB)**
   * Stores persistent data, such as user details, orders, and inventory.

## 2. Key Workflows

### Workflow 1: User Browses Books

1. User navigates to the bookstore homepage.
2. Frontend sends a request to /books endpoint.
3. Backend retrieves book data from MongoDB or Sanity CMS and responds with book listings.
4. Frontend displays the data to the user.

### Workflow 2: User Adds Books to Cart

1. User selects a book and clicks "Add to Cart."
2. Frontend sends a POST request to /cart with book details.
3. Backend updates the user’s cart in MongoDB.
4. Frontend updates the cart icon to reflect the added book.

### Workflow 3: User Places an Order

1. User reviews their cart and clicks "Checkout."
2. Frontend sends a POST request to /orders with cart details and payment information.
3. Backend validates the order, processes the payment, and updates the database.
4. User receives an order confirmation email.

## 3. Category-Specific Instructions

### General eCommerce Features for Bookstore

1. **Product Browsing:**
   * Use /books endpoint to fetch available books.
   * Include filters for categories, authors, and price range.
2. **Cart Management:**
   * Endpoints for adding, updating, and removing items:
     + POST /cart
     + PATCH /cart
     + DELETE /cart
3. **Order Placement:**
   * Use /orders endpoint to create and track orders.
   * Integrate payment gateways such as Stripe.

## 4. API Endpoints

| **Endpoint** | **Method** | **Purpose** | **Response Example** |
| --- | --- | --- | --- |
| /books | GET | Fetches all book listings | [{"id":1,"name":"Book A","price":100}] |
| /book/:id | GET | Fetch details of a specific book | {"id":1,"name":"Book A","author":"Author A"} |
| /cart | POST | Adds a book to the cart | {"success":true} |
| /cart | PATCH | Updates quantity of a book in the cart | {"success":true} |
| /cart | DELETE | Removes a book from the cart | {"success":true} |
| /orders | POST | Places an order | {"orderId":123} |
| /orders/:id | GET | Fetch details of a specific order | {"id":123,"status":"Processing"} |

## 5. Sanity Schema Example

### Product Schema for Books

export default {

name: 'book',

type: 'document',

fields: [

{ name: 'title', type: 'string', title: 'Title' },

{ name: 'author', type: 'string', title: 'Author' },

{ name: 'price', type: 'number', title: 'Price' },

{ name: 'stock', type: 'number', title: 'Stock Level' },

{ name: 'category', type: 'string', title: 'Category' },

{ name: 'isbn', type: 'string', title: 'ISBN' },

],

};

### Order Schema

export default {

name: 'order',

type: 'document',

fields: [

{ name: 'userId', type: 'string', title: 'User ID' },

{ name: 'books', type: 'array', of: [{ type: 'reference', to: [{ type: 'book' }] }], title: 'Books' },

{ name: 'total', type: 'number', title: 'Total Price' },

{ name: 'status', type: 'string', title: 'Status' },

],

};

## 6. Technical Roadmap

### Milestones and Deliverables

1. **Week 1:**
   * Finalize system architecture and data schema.
   * Set up Sanity CMS and MongoDB.
2. **Week 2:**
   * Implement API endpoints for book browsing and cart management.
   * Design frontend components for product listing and cart.
3. **Week 3:**
   * Integrate payment gateway and order placement workflows.
   * Test end-to-end user flows.
4. **Week 4:**
   * Conduct performance and security testing.
   * Prepare documentation and deploy the application.

## 7. Collaborative Feedback

### Peer Review Notes

* Ensure API error handling for scenarios like out-of-stock books.
* Enhance user experience with features like wishlists and book recommendations.

### Scalability Suggestions

* Use Redis for caching frequently accessed data like book categories.
* Implement rate-limiting to handle high traffic during promotions.

## Submission Requirements

1. **Repository Submission:**
   * Include the documentation in a folder named Documentation.
2. **File Naming Convention:**
   * SystemArchitecture\_Bookstore.pdf
   * APIEndpoints.xlsx
   * SanitySchema.js
3. **Review and Quality Check:**
   * Validate diagrams and schemas for clarity and completeness.
   * Incorporate feedback from peers and mentors.

## What’s Next?

* Prepare for Day 3 by implementing CRUD operations in Sanity CMS for books and orders.
* Start integrating the API with the frontend for a seamless user experience.

This document serves as a blueprint for the Bookstore Marketplace, ensuring clarity and professionalism in its development and presentation.