***Day: 02 Planning The Technical Documentation***

**Project Name: Shop.Co (E-commerce Platform for Clothing)**

**Developed By: SANA NAZ**

* **Recap: Business Focus**

\* **Goals**: Address limited online access to affordable, trendy clothing.

\* **Target** Audience: Young adults seeking stylish, budget-friendly fashion.

\* **UVP**: Personalized recommendations and exclusive discounts.

\* **Data Schema**: Drafted entities (Products, Orders, Customers) and relationships.

**2. Technical Requirements**

**Frontend Requirements:**

* User-friendly interface for browsing products.
* Responsive design for mobile and desktop devices.
* Essential pages: Home, Product Listing, Product Details, Cart, and Order Confirmation.

**Backend Requirements:**

* Use Sanity CMS for managing product, order, and customer data.
* Design custom schemas in Sanity aligned with business objectives.

**Third-Party Integrations:**

* Clerk for authentication.
* APIs for shipment tracking and payment processing.

**Technologies Used:**

**- Frontend Framework: Next.js (for server-side rendering and routing)**

**- CSS Framework: Tailwind CSS (for responsive and modern styling)**

**- Backend: Sanity CMS (content management for product data)**

**- State Management: useState, useEffect (React hooks)**

**- Deployment: Vercel (for hosting)**

**3. System Architecture**

**Overview**: The system consists of a modular, component-based frontend and a CMS-backed data management system.

***Frontend:***

* Next.js for page routing and server-side rendering.
* Components: Navbar, Footer, Home, ProductList, ProductDetail, Cart, Checkout
* Pages: `/`, `/products`, `/cart`, `/checkout`
* Navigation: Uses Next.js Link for page transitions.

***Backend:***

* Sanity CMS: Manages product inventory, categories, and order details.
* Fetching data: `getStaticProps` and `getServerSideProps` are used to fetch data during page rendering.

***Third-Party Services:***

* Clerk for user authentication.
* APIs for mock payment and shipment tracking.

**Key Workflows:**

* User browses products -> Data fetched from Sanity.
* User places an order -> Order saved in Sanity.
* Payment processed -> Status updated via mock API.

**4. API Requirements**

|  |  |  |
| --- | --- | --- |
| **Endpoint** | **Method** | **Description** |
| /api/products | GET | Retrieve all product details |
| /api/product/:id | GET | Retrieve a single product by ID |
| /api/cart | POST | Add items to the shopping cart |
| /api/checkout | POST | Submit order and customer information |

**5. Technical Documentation**

* **System Architecture**: Include an Excalidraw diagram showing frontend-backend integration.
* **API Documentation**: List all endpoints with methods, payloads, and sample responses.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Endpoint** | **Method** | **Description** | **Payload** | **Response Example** |
| /api/products | GET | Fetches all product details | - | { "id": 1, "name": "T-shirt", "price": 25 } |
| /api/product/:id | GET | Fetches a single product by ID | - | { "id": 1, "name": "T-shirt", "price": 25 } |
| /api/cart | POST | Adds items to the shopping cart | { "productId": 1, "quantity": 2 } | { "cartId": 123, "status": "Item added" } |
| /api/checkout | POST | Submits order and customer details | { "customer": { "name": "John Doe", "email": "john@example.com" }, "cart": [{ "productId": 1, "quantity": 2 }] } | { "orderId": 456, "status": "Order confirmed" } |
| /api/shipment | GET | Retrieves order shipment status | { "orderId": 456 } | { "orderId": 456, "status": "In Transit" } |

* **Data Schema**:

export default {

name: 'product',

type: 'document',

fields: [

{ name: 'name', type: 'string', title: 'Product Name' },

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

{ name: 'size', type: 'string', title: 'Size' },

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

{ name: 'image', type: 'image', title: 'Product Image' }

]

};

* **Workflow Descriptions**: Detail key interactions, like adding products to a cart or tracking orders.