# Marketplace Technical Foundation [Furniture]

### Define Technical Requirments:

#### Frontend requirements:

A frontend will deliver a user -friendly experience to with the following pages and options.

###### Home page

It displays featured categories, popular products, shop now option , flash sales and cart option.

###### ii. Products

Products will display with options to filter by price or brands.

###### iii. Add to cart

When user adds the product into cart which will update dynamically with quantity and price.

###### iv. Payment

User will allowed with multiple options of payment which he can select at time of buying.

###### v. Checkout

It will display shipping details, payment methods and order summary.

###### vi. Order confirmation

A pop-up message will be shown to the user which contain order , product details and estimated delivery time .

#### Sanity CMS:

It acts as a backend to manage product data , user info, and order records.

###### Product schema:

* Fields

Product Id

Name

Price

Stock

Description

Image

###### Order schema:

* Fields

Order Id

Product Id

Price

Quantity

Delivery charges

Total amount

###### Customer schema:

* Fields

Customer Id

Name

Contact No

Email

Address

###### Payment schema:

* Fields

Order Id

Payment method

Payment status

###### Shipment schema:

* Fields

Shipment Id

Order Id

Courier service

Tracking no

Delivery date

#### Third-Party API:

To make marketplace more dynamically and add more functions we will use third-party APIs for payments and shipments.

###### Payment gateways:

Multiple options will provide to users like easypaisa or debit , credit card services through which he can make payment securely.

###### Shipment tracking API:

Shipment options will be added for users to track there products and get product on there door steps like ship engine.

### Design System Architecture

#### User:

A user visit on landing page via mobile or computer on furniture marketplace. The request will move forward to product data API.

#### Product display:

The product data fecthes data from sanity and displays the data dynamically to the user.

#### Order placement:

When the user place order , all details regarding user info and product details are sent to sanity and store there.

#### Shipment tracking:

Shipment tracking information is fetched through third party API which reflects to the user.

#### Payment process:

Payment details are securely processed through the payment gateway. Then confirmation is sent to the user and recorded in sanity.

# Data Flow Example

User

Frontened

(Nextjs)

Order Placing

API

(Sanity)

Product Data

Third-party API

Shipment

Tracking

Payment Process

### Plan API Requirements

Here is a table summarizing the data API endpoints and descriptions

|  |  |  |  |
| --- | --- | --- | --- |
| **Endpoints** | **Method** | **Description** | **Response example** |
| /api/products | GET | Fetch all products | {  “productId” : 1,  “name” : “BlueSofa”,  “price” : 24000,  “stock” : 20,  “image” : “url”  } |
| /api/orders | POST | Create a new order in sanity | {  “orderId” : 1234,  “productId” : 1,  “quantity” : 2,  “price” : 48000,  “deliverCharges” : 2000,  } |
| /api/shipment | GET | Track order status via third-party API | {  “orderId” : 1234,  “shipmentId” : 5432,  “trackingNo” : #7862,  “deliveryDate” : “dd/mm/yyyy”,  “deliverCharges” : 2000,  } |

### Schema example:

### export default {

### // Define the document type and its name

### name: 'product',

### type: 'document',

### title: 'Product', // The title displayed in the CMS for this document

### fields: [

### {

### // Field for the product's name

### name: 'name',

### type: 'string', // Basic text field

### title: 'Product Name', // Label for the field in the CMS

### validation: (Rule) =>

### Rule.required},

### {

### // Field for a detailed description of the product

### name: 'description',

### type: 'text', // Multi-line text field

### title: 'Description',

### description: 'Detailed description of the product.',

### validation: (Rule) =>

### Rule.required()

### },

### {

### // Field for the product's price

### name: 'price',

### type: 'number', // Numeric field

### title: 'Product Price',

### validation: (Rule) =>

### Rule.required()

### },

### {

### // Field for the discount percentage

### name: 'discountPercentage',

### type: 'number',

### title: 'Discount Percentage',

### description: 'Percentage discount on the product.'

### },

### {

### // Field for the price before the discount

### name: 'priceWithoutDiscount',

### type: 'number',

### title: 'Price Without Discount',

### description: 'Original price of the product before discount.',

### readOnly: true, // Makes this field non-editable in the CMS

### },

### {

### // Field for the product's rating

### name: 'rating',

### type: 'number',

### title: 'Rating',

### description: 'Average rating of the product.',

### validation: (Rule) =>

### Rule.min(0) // Ensures the rating is not negative

### .max(5) // Limits the rating to a maximum of 5

### .error('Rating must be between 0 and 5.'),

### },

### 

### 

### {

### // Field for the product image

### name: 'image',

### type: 'image',

### title: 'Product Image',

### description: 'High-quality image of the product.',

### options: {

### hotspot: true, // Enables cropping and focal point selection

### },

### },

### ],

### };