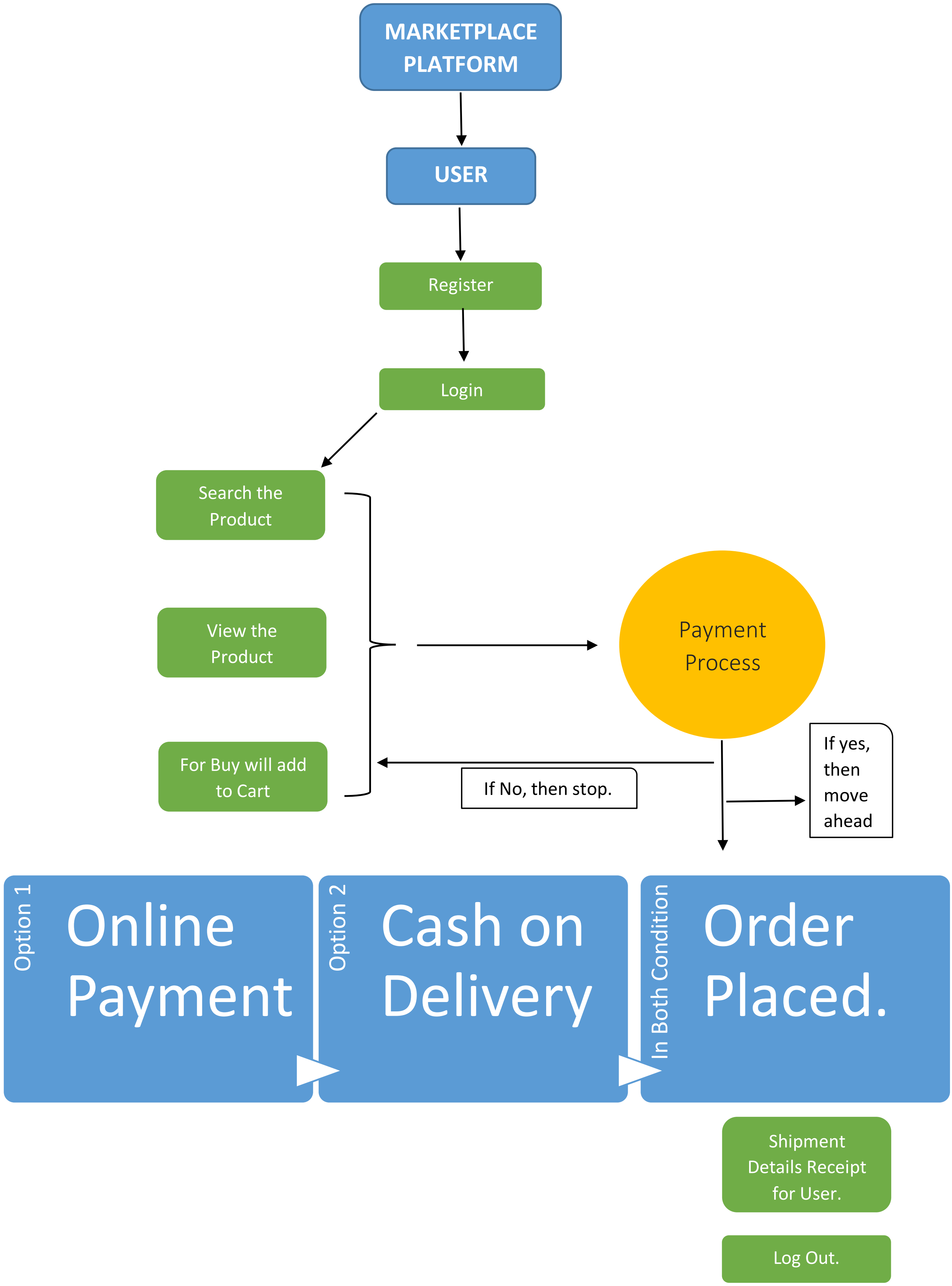
FRONTEND REQUIRED FLOW CHART



MARKETPLACE PLATFORM

USER

Register

Login

Search the Product

View the Product

For Buy will add to Cart

Payment Process

If No, then stop.

If yes, then move ahead

Option 1 Online Payment

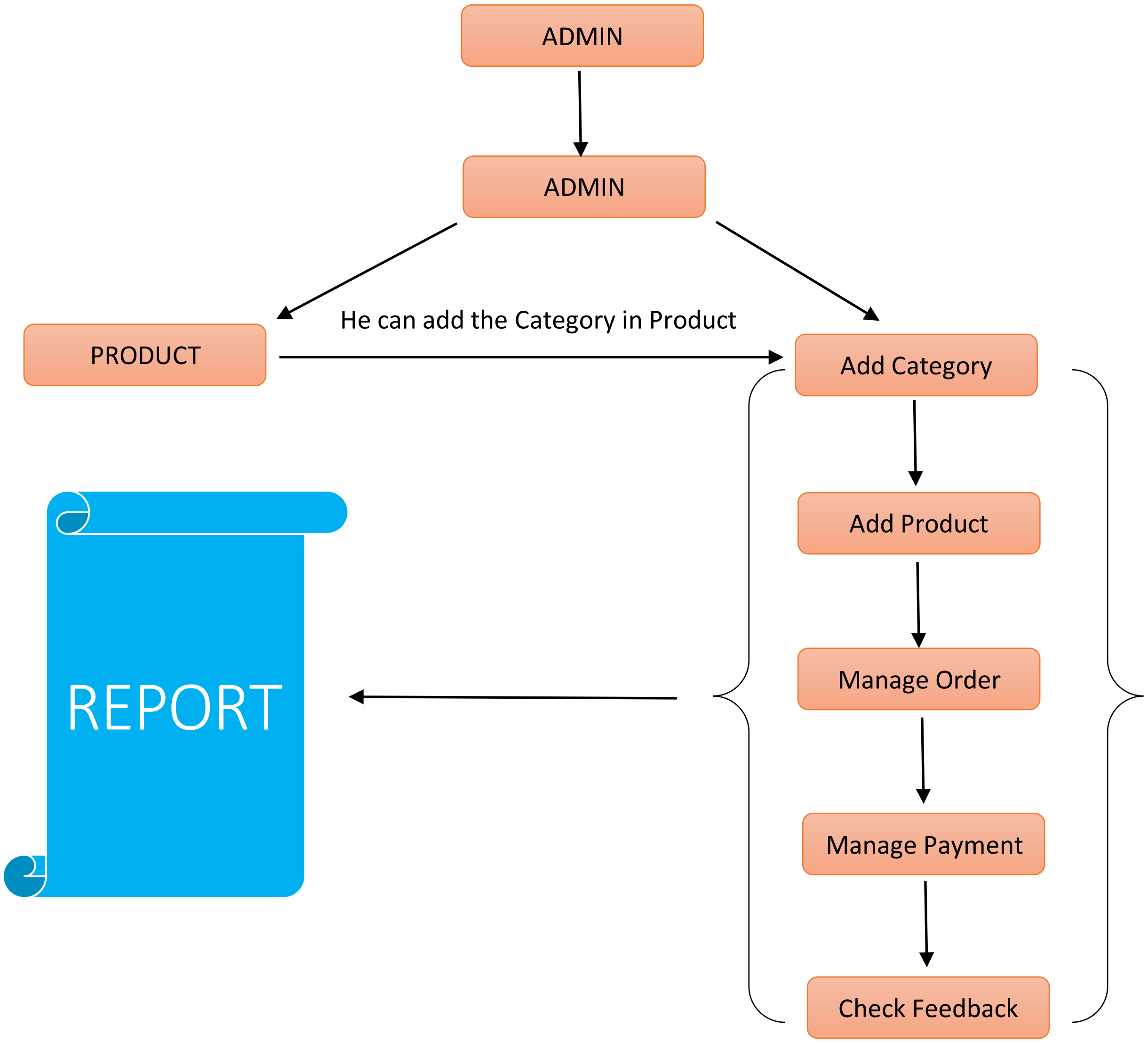
Option 2 Cash on Delivery

In Both Condition Order Placed.

Shipment Details Receipt for User.

Log Out.

ADMIN FLOW CHART



ADMIN

ADMIN

PRODUCT

He can add the Category in Product

Add Category

Add Product

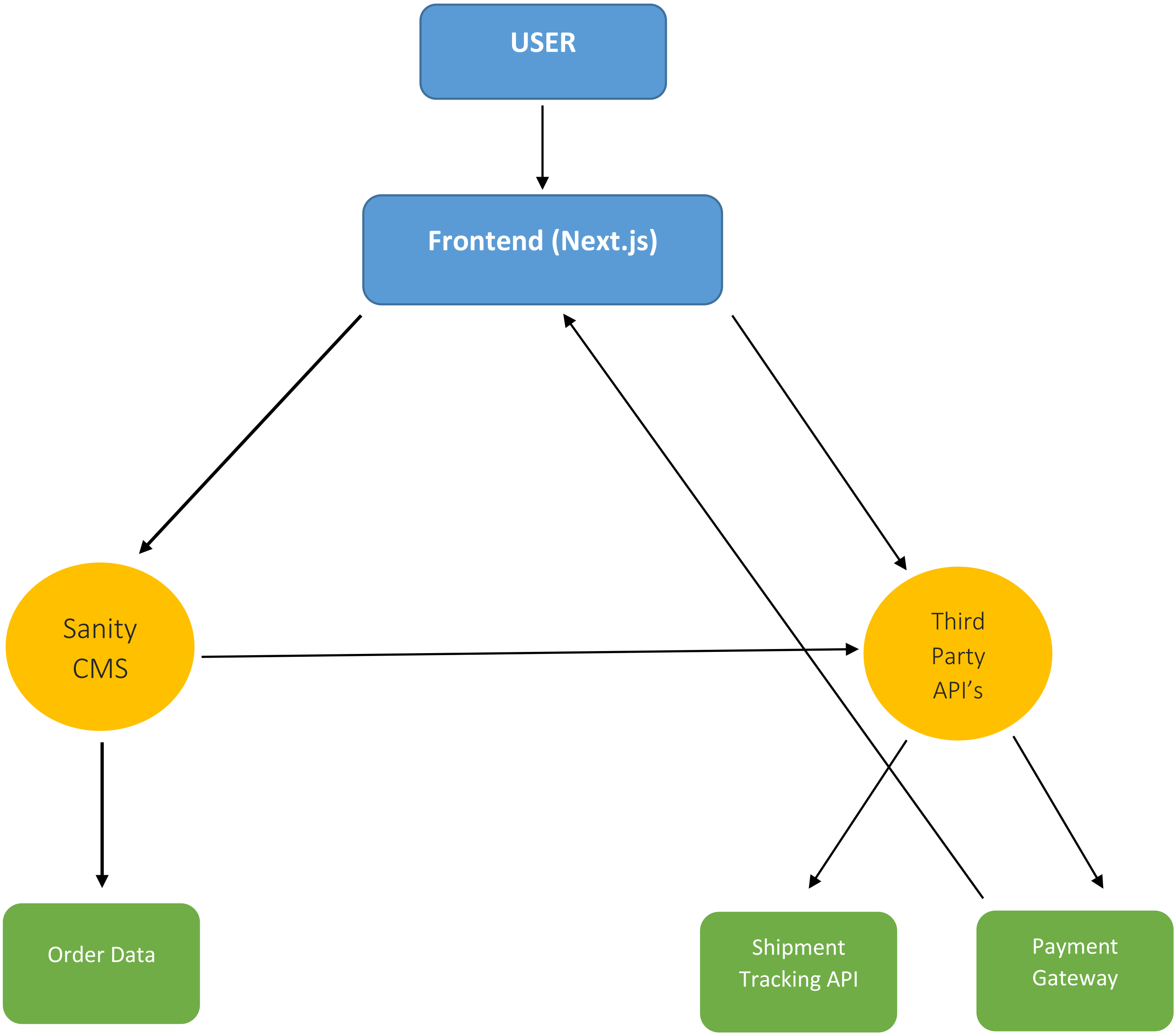
Manage Order

Manage Payment

Check Feedback

REPORT

Planning the Technical Foundation Diagram



EXPLANATION

User	USER WILL PERFROM THE ACTION ON FRONTEND PAGE LIKE PRODUCT BROWSING, SELECT THE PRODUCT AND PLACING THE ORDER.
FRONTEND (NEXT.JS)	SEND USER ACTION TO THE BACKEND AND THIRD PARTY API'S.
SANITY (CMS)	MANAGE AND STORE ALL ORDER AND PRODUCT DATA.
THIRD PARTY API'S	THIS WILL HANDLE THE SHIPMENT TRACKING AND PAYMENT PROCESSORS.
SHIPMENT TRACKING	THIS WILL PROVIDE REAL TIME TRACKING UPDATES TO THE FRONTEND FOR USER.
PAYMENT GATEWAY	WILL PROCESS THE PAYMENT AND WILL SEND BACK THE CONFIRMATION UPDATE.
ORDER DATA	ALL THESE DATA WILL STORED IN SANITY CMS FOR BACKEND REFRENCE. (HISTORY).

Transitioning to Technical Planning.

Define Technical Requirement in any Platform.

Frontend Requirement.

1- Make sure that The Platform should be user-friendly for browsing the product and also should be responsive on Mobile or Desktop Device.

Essential Pages.

2- Following pages should be part of your platform for the user interaction.

- a- **Home**
- b- **About**
- c- **Menu**
- d- **Shop**
- e- **Category**
- f- **Sign In**
- g- **Sing Up**
- h- **Contact**
- i- **Product ID (Dynamic Page)**
- j- **Add to Cart**
- k- **Continuous Shopping**
- l- **Payment Method**
- m- **Shipment Tracking (Status)**
- n- **Order Confirmation.**
- o- **Check Out**

Backend Requirement With Sanity (CMS).

3- Use Sanity (CMS) to managing the following:

- a- **Product Data, Order Details, History Records of user orders.**
- b- **Draw Schemas in Sanity to align with product structure (e.g., name, price, stock, id, etc…) &**
- c- **also for Order structure (e.g., customer info, order details, payment details)**

Third Party API's.

4- This will handle the following tasks after placing the order.

- a. **Shipment tracking information is fetched through a Third-Party API.**
- b. **Real-time tracking updates are displayed to the user.**
- c. **Payment details are securely processed through the Payment Gateway.**
- d. **A confirmation is sent back to the user and recorded in Sanity CMS.**

Interfaces (Data Models):

Product:

- Fields: _id, name, price, category, stock, image, etc.

Customer:

- Fields: _id, name, email, address, phoneNumber, etc.

Order:

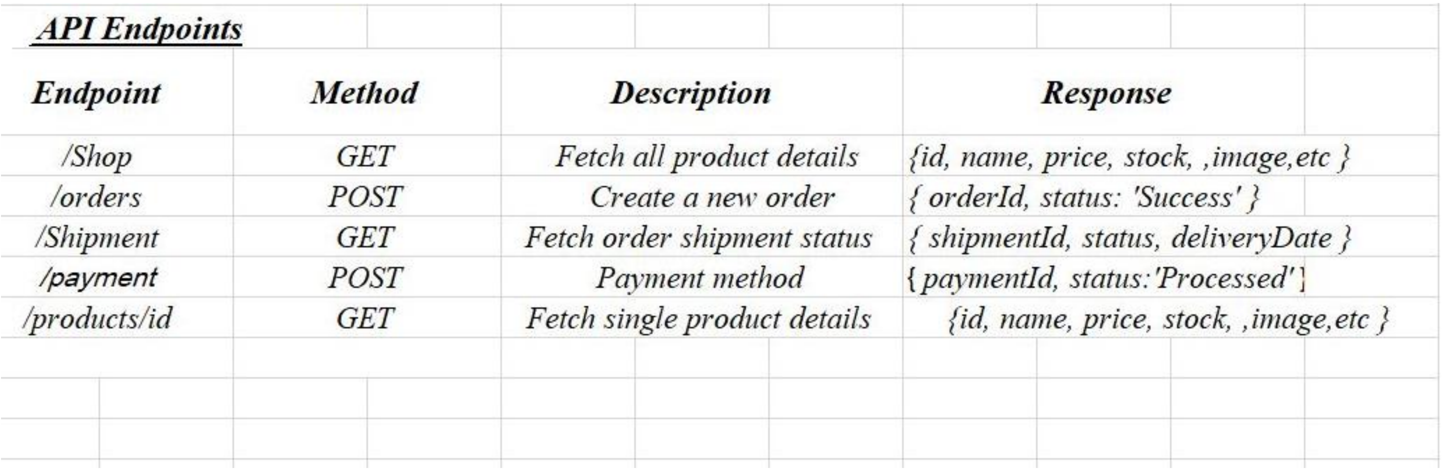
- Fields: _id, productID, userID, orderStatus, totalAmount, etc.

Payment:

- Fields: _id, orderID, paymentMethod, amount, transactionDate, etc.

Shipment:

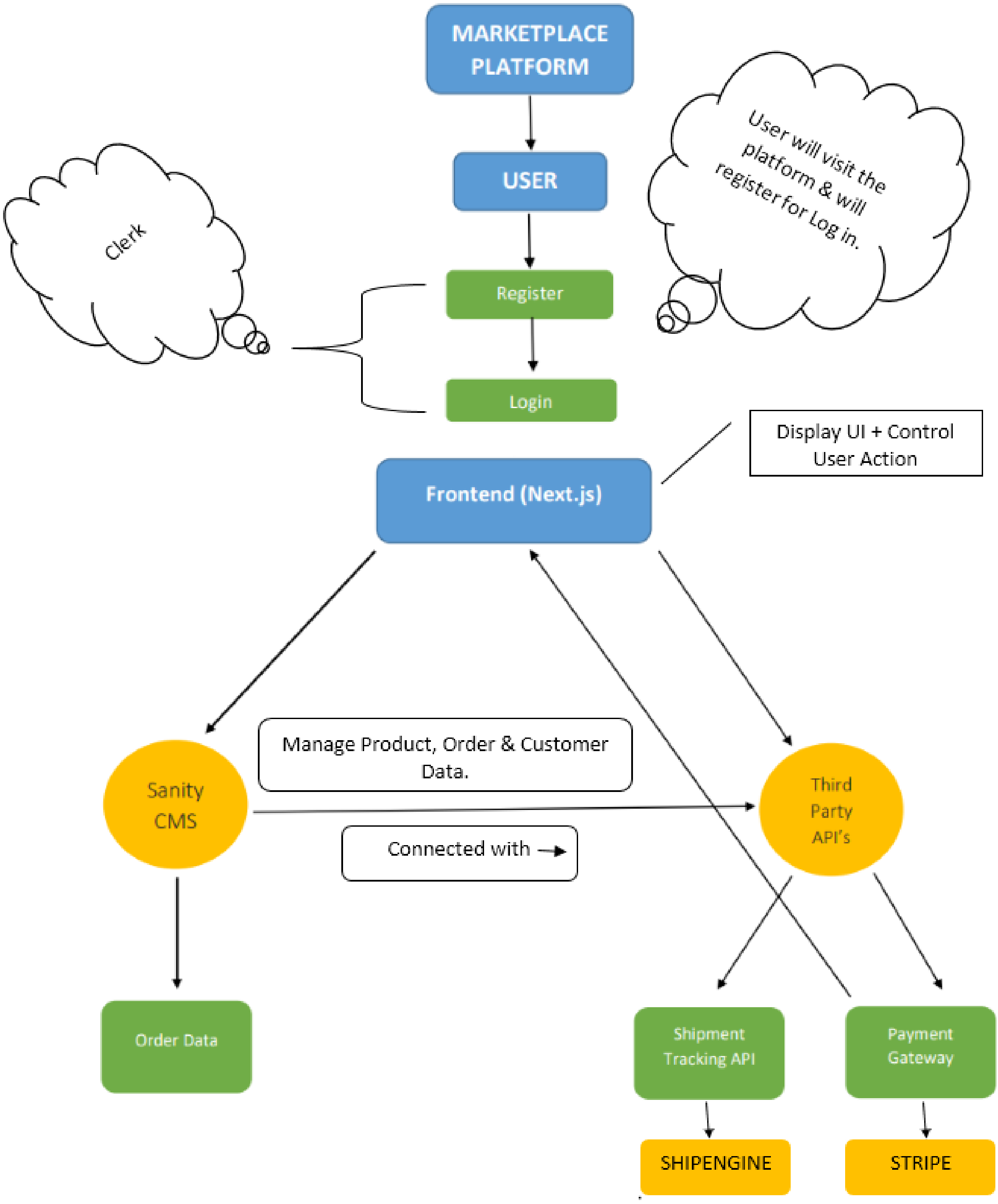
- Fields: _id, orderID, courierDetails, paymentStatus, estimatedDeliveryDate.



API Endpoints

Endpoint	*Method*	*Description*	*Response*
/Shop	GET	Fetch all product details	{id, name, price, stock, ,image,etc }
/orders	POST	Create a new order	{ orderId, status: 'Success' }
/Shipment	GET	Fetch order shipment status	{ shipmentId, status, deliveryDate }
/payment	POST	Payment method	{ paymentId, status:'Processed' }
/products/id	GET	Fetch single product details	{id, name, price, stock, ,image,etc }

DESIGN SYSTEM ARCHITECTURE

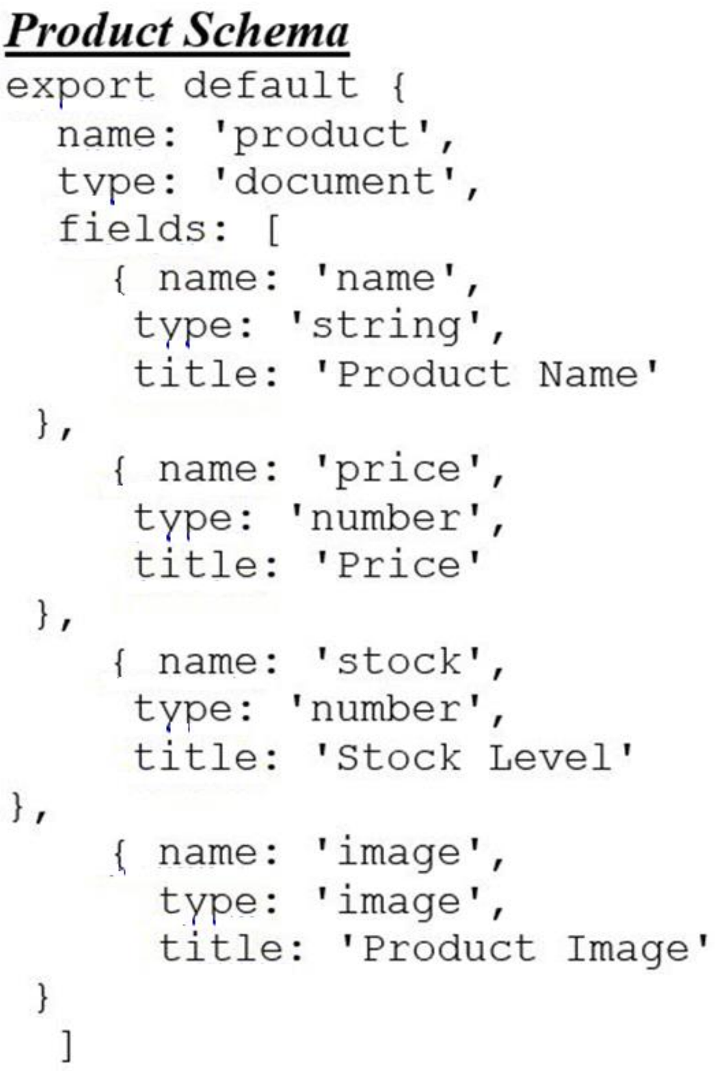


EXPLANATION

User	USER WILL PERFROM THE ACTION ON FRONTEND PAGE LIKE PRODUCT BROWSING, SELECT THE PRODUCT AND PLACING THE ORDER.
FRONTEND (NEXT.JS)	SEND USER ACTION TO THE BACKEND AND THIRD PARTY API'S.
SANITY (CMS)	MANAGE AND STORE ALL ORDER AND PRODUCT DATA.
THIRD PARTY API'S	THIS WILL HANDLE THE SHIPMENT TRACKING AND PAYMENT PROCESSORS.
SHIPMENT TRACKING	THIS WILL PROVIDE REAL TIME TRACKING UPDATES TO THE FRONTEND FOR USER.
PAYMENT GATEWAY	WILL PROCESS THE PAYMENT AND WILL SEND BACK THE CONFIRMATION UPDATE.
ORDER DATA	ALL THESE DATA WILL STORED IN SANITY CMS FOR BACKEND REFRENCE. (HISTORY).

Sanity Schema Example

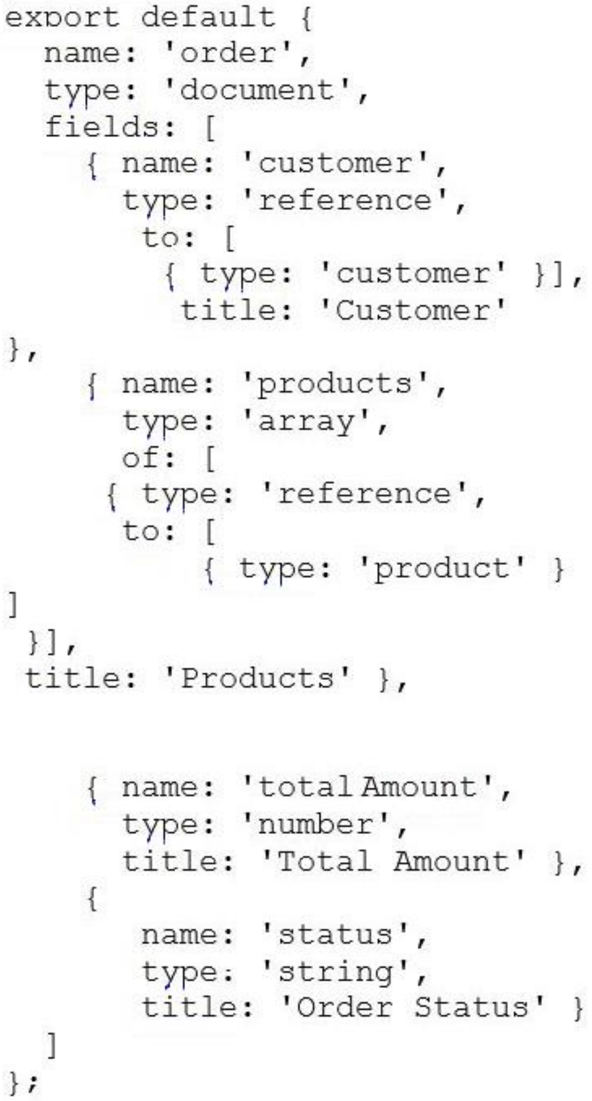
Here’s an example schema for managing product categories and products in Sanity.io:



Product Schema

```
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'
 }
  ]
```

Order Schema



```
export default {
  name: 'order',
  type: 'document',
  fields: [
    { name: 'customer',
      type: 'reference',
       to: [
        { type: 'customer' }],
         title: 'Customer'
},
    { name: 'products',
      type: 'array',
      of: [
     { type: 'reference',
      to: [
          { type: 'product' }
]
 }],
 title: 'Products' },


    { name: 'total Amount',
      type: 'number',
      title: 'Total Amount' },
    {
       name: 'status',
       type: 'string',
       title: 'Order Status' }
  ]
};
```