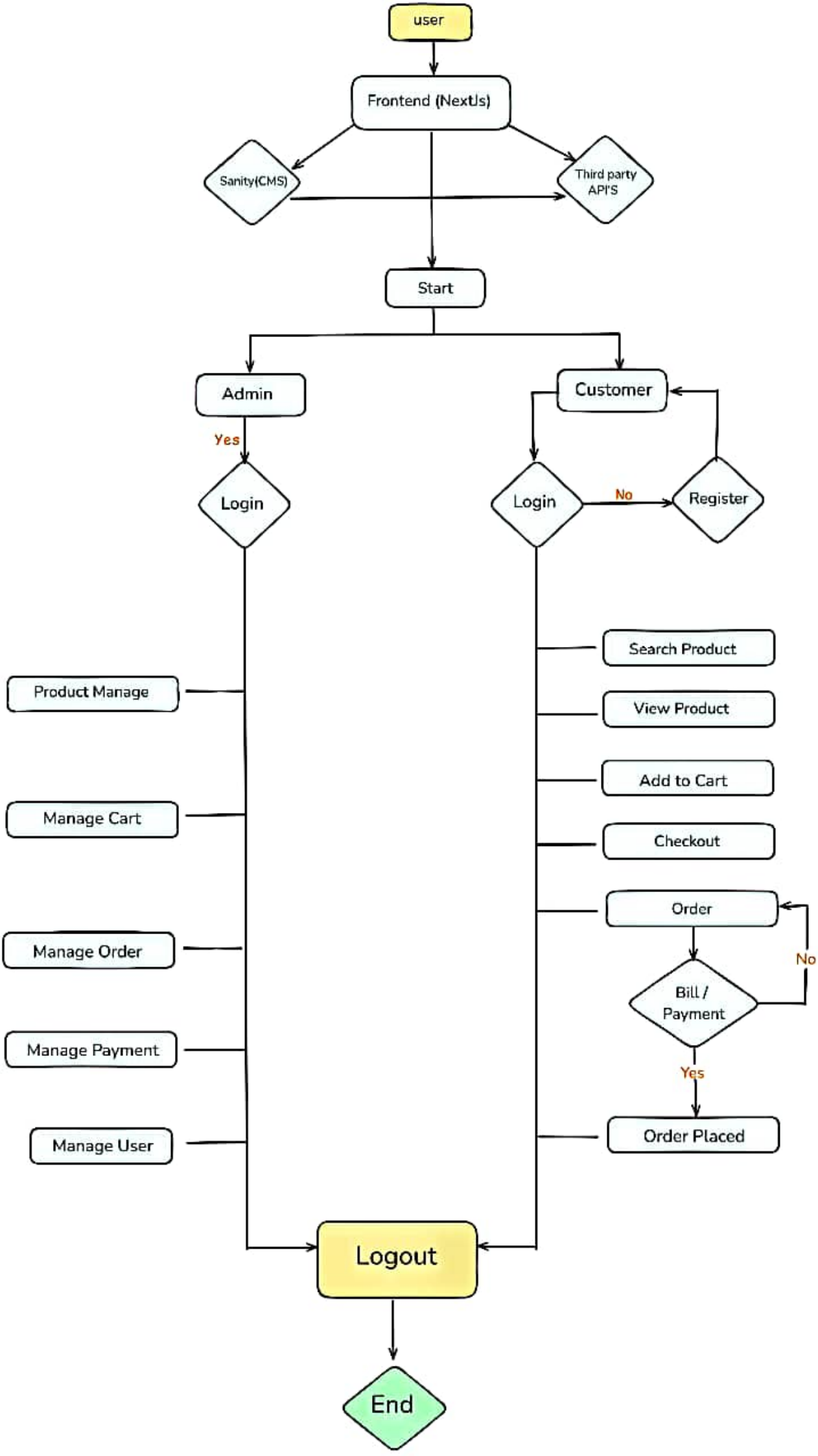
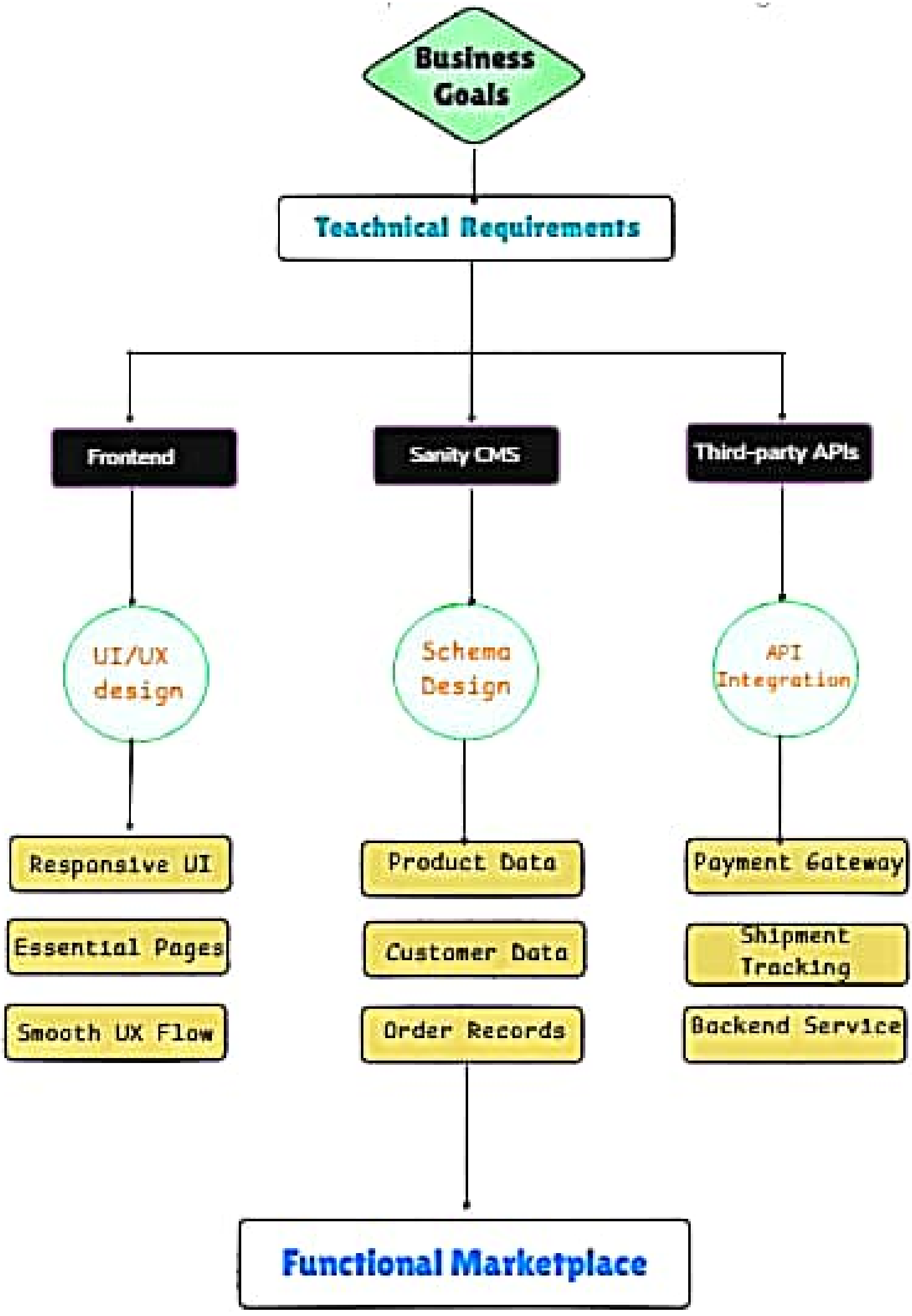

user
Frontend (NextJs)
Sanity(CMS)
Third party API'S
Start
Admin
Yes
Login
Customer
Login
No
Register
Product Manage
Manage Cart
Manage Order
Manage Payment
Manage User
Search Product
View Product
Add to Cart
Checkout
Order
Bill / Payment
No
Yes
Order Placed
Logout
End

COMPONENT	DESCRIPTION
Frontend	Displays products, cart, and checkout pages. Sends user actions to the backend.
Backend	Processes API requests, communicates with Sanity CMS, and handles business logic.
Sanity CMS	Stores product details, categories, and blog content. The backend fetches data from Sanity.
Third Party APIs	Tracks order delivery status and updates users.
Payment Gateway	Processes customer payments securely.

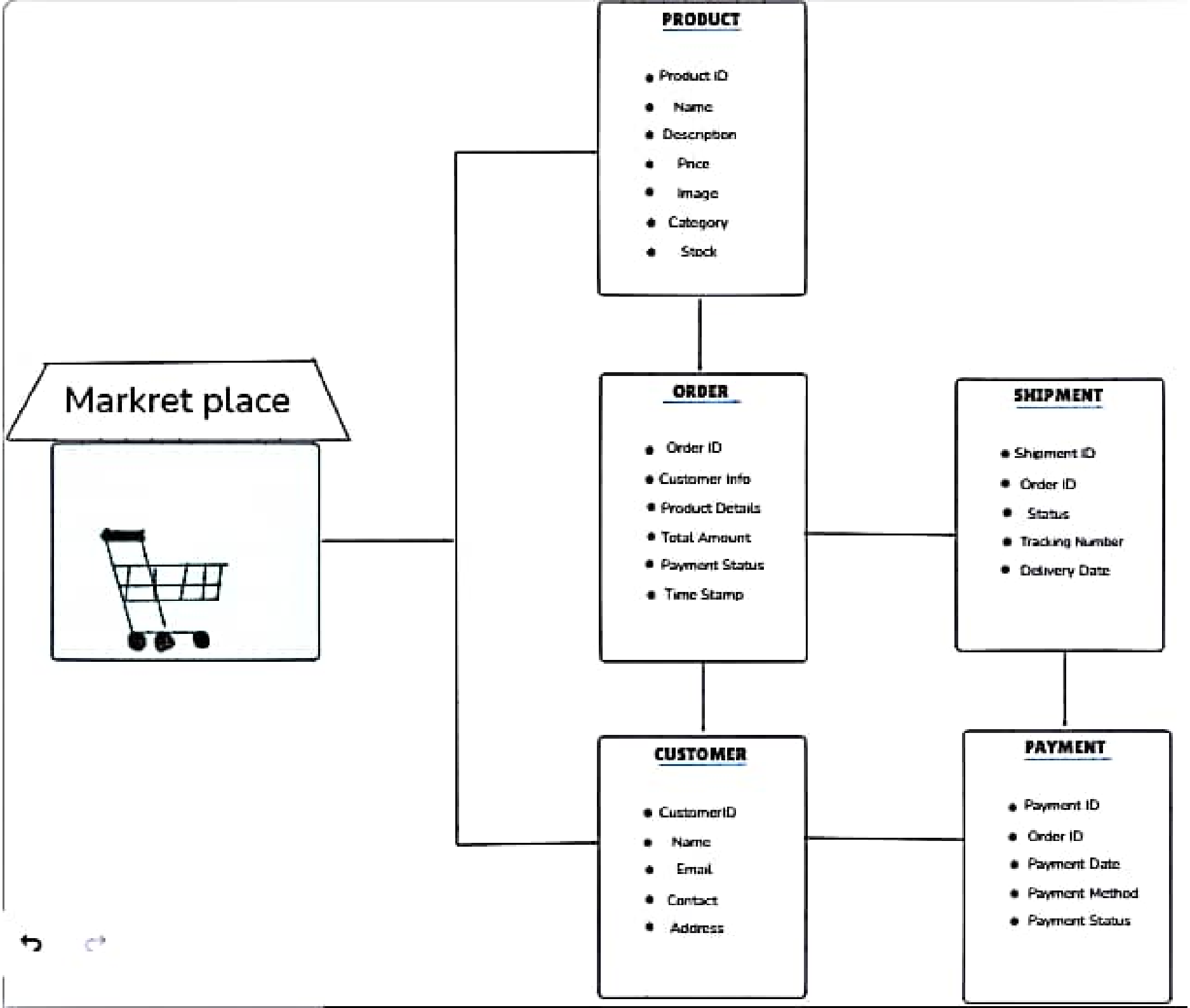
COMPONENT INTERACT

- Frontend requests product data → Backend fetches from Sanity CMS.
- User adds products to cart → Cart data is stored temporarily.
- Order is placed → Backend saves order details in the database.
- Payment is processed → Sent to Stripe/PayPal.
- Order is shipped → Backend updates tracking using the Shipping API.
- User tracks shipment → Backend fetches updates from the Shipping API.


Business Goals
Teachnical Requirements
Frontend
Sanity CMS
Third-party APIs
UI/UX design
Schema Design
API Integration
Responsive UI
Essential Pages
Smooth UX Flow
Product Data
Customer Data
Order Records
Payment Gateway
Shipment Tracking
Backend Service
Functional Marketplace

Step: 1 Business Golas in Technical Requirements

- The business goals guide the overall project.
- These are broken down into three key areas : .
  - Frontend (UI/UX, responsive design, essential pages).
  - Sanity CMS (schema for products, customers, orders).
  - Third-Party APIs (payment, shipping, backend services).
- Each section ensures a functional and scalable marketplace.


PRODUCT
Product ID
Name
Description
Price
Image
Category
Stock
Markret place
ORDER
Order ID
Customer Info
Product Details
Total Amount
Payment Status
Time Stamp
SHIPMENT
Shipment ID
Order ID
Status
Tracking Number
Delivery Date
CUSTOMER
CustomerID
Name
Email
Contact
Address
PAYMENT
Payment ID
Order ID
Payment Date
Payment Method
Payment Status

PRODUCT API

Entities	Method	Description	Request Body	Response
Products	Get	Fetch all available products	None	List of products with details
Products	Get	Fetch a specific product by ID	None	Product details
Products	Get	Add a new product	{ "name": "Sofa", "description": "Luxury sofa", "price": 500, "stock": 10, "image_url": "link", "category_id": 2 }	{ "message": "Product created", "product_id": 10 }

PAYMENT API

Entities	Method	Description	Request Body	Response
Payment	Post	Process payment for an order	{ "order_id": 101, "payment_method": "Credit Card", "amount": 350 }	{ "message": "Payment successful", "payment_id": 5001, "payment_status": "Completed"}
Payment	Get	Get payment status	None	{ "order_id": 101, "payment_status": "Completed" }

CUSTOMER API

Entities	Method	Description	Request Body	Response
Customer	Post	Register a new customer	{ "name": "John", "email": "john@example.com", "phone": "1234567890", "address": "Street, City" }	{ "message": "Customer registered", "customer_id": 5 }
Customer	Get	Fetch customer details by ID	None	Customer details

ORDER API

Entities	Method	Description	Request Body	Response
Order	Post	Place a new order	{ "customer_id": 5, "items": [{"product_id": 1, "quantity": 2}], "total_price": 350, "status": "Pending" }	{ "message": "Order placed", "order_id": 101 }
Order	Get	Fetch order details	None	Order details with items

SHIPMENT API

Entities	Method	Description	Request Body	Response
Shipment	Get	Track order shipment	None	{ "shipment_id": 2001, "tracking_number": "TRK123456789", "status": "In Transit" }