Technical Requirements

🔹 Frontend (Next.js)

The user-facing interface where customers browse products, add to cart, and place orders.

- ❖ **Pages:**
  - ➢ Home (restaurant intro, featured items)
  - ➢ About
  - ➢ Sign in / Sign up
  - ➢ Contact
  - ➢ Product Listing/Menu (all available dishes)
  - ➢ Product Details (single product info)
  - ➢ Cart (items added before checkout)
  - ➢ Checkout (payment & delivery details)
  - ➢ Order Confirmation (receipt & tracking info)
- ❖ **Technologies:**
  - ➢ **Frontend** (for server-side rendering & routing)
  - ➢ **React** (component-based UI)
  - ➢ **Tailwind CSS / Shadcn UI**

🔹 Backend (Sanity CMS)

Sanity CMS will act as the database to manage:

- ❖ **Products** (name, description, price, category, images)
- ❖ **Orders** (order ID, customer details, status)
- ❖ **Users** (registration, authentication)
- ❖ **Payments** (payment ID, order details, status)
- ❖ **SLA Tracking**
- ❖ **Technologies:**
  - ➢ **Sanity CMS** (headless CMS for content storage)
  - ➢ **Sanity API** (to fetch data for frontend)

🔹 Third-Party APIs

To handle additional features:

- **Payment Processing**
- **Shipment Tracking**

🔹 State Management

For handling cart, user sessions, and order tracking.

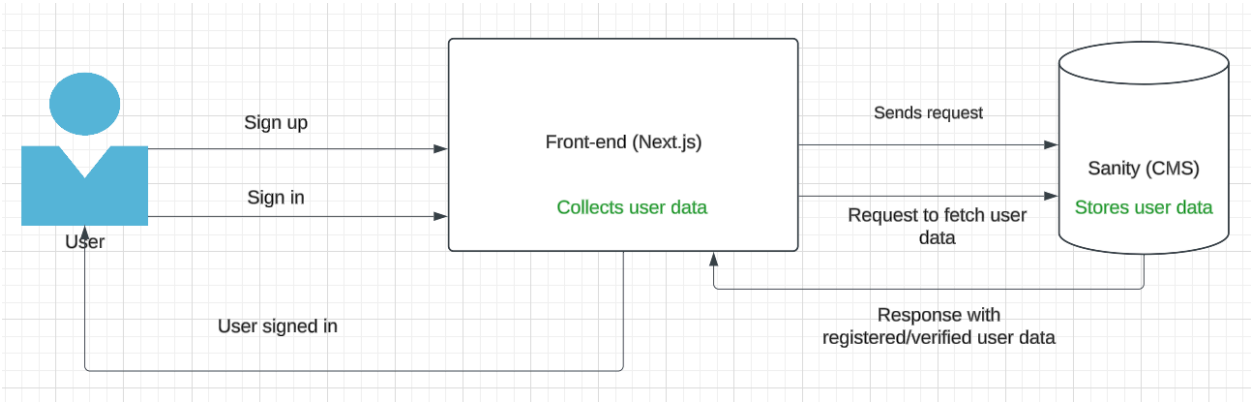
- **React Context API** (lightweight)
- **Redux** (if complex state management is needed)

❖ **Overall Technologies:**

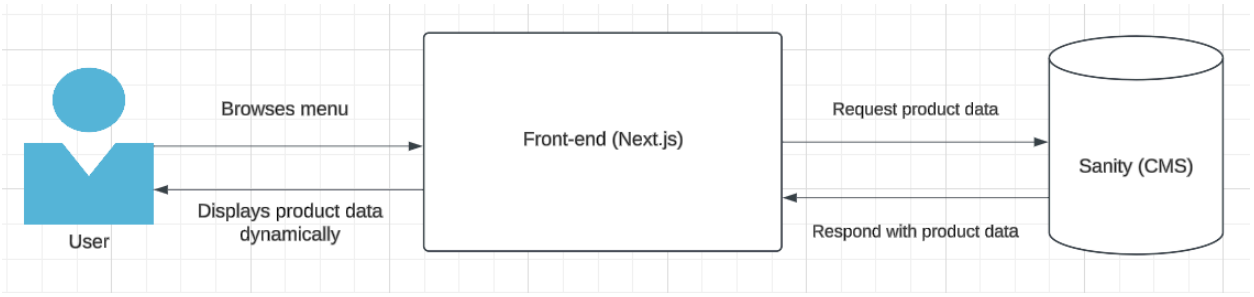
- ➢ **Frontend** → Next.js, Tailwind CSS
- ➢ **Backend** → Sanity CMS
- ➢ **Database** → Sanity CMS
- ➢ **APIs** → Payment API, Shipment API,React Context API
- ➢ **Authentication** → OAuth/Firebase/Auth0/Nextauth
- ➢ **Communication** → Webhooks (for real-time updates)

System Architecture

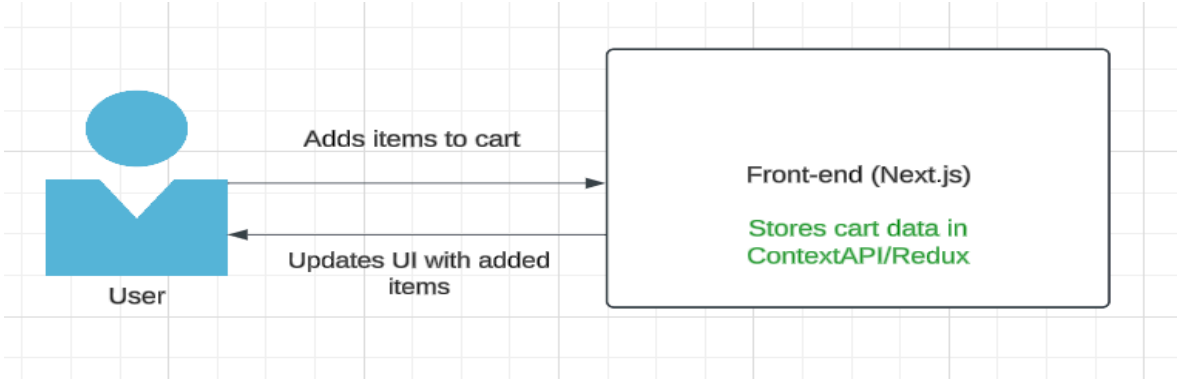
1- User Registration



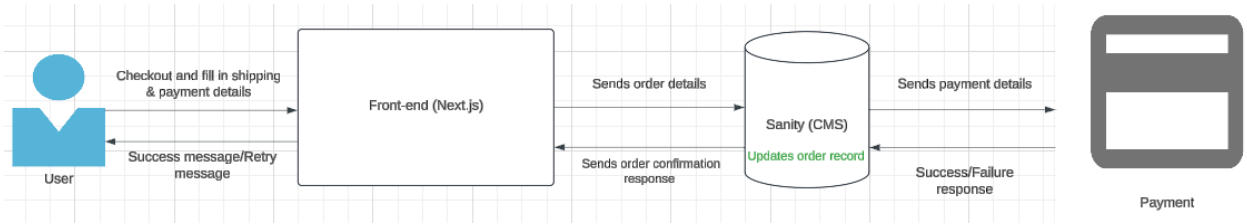
2- User Browses Products



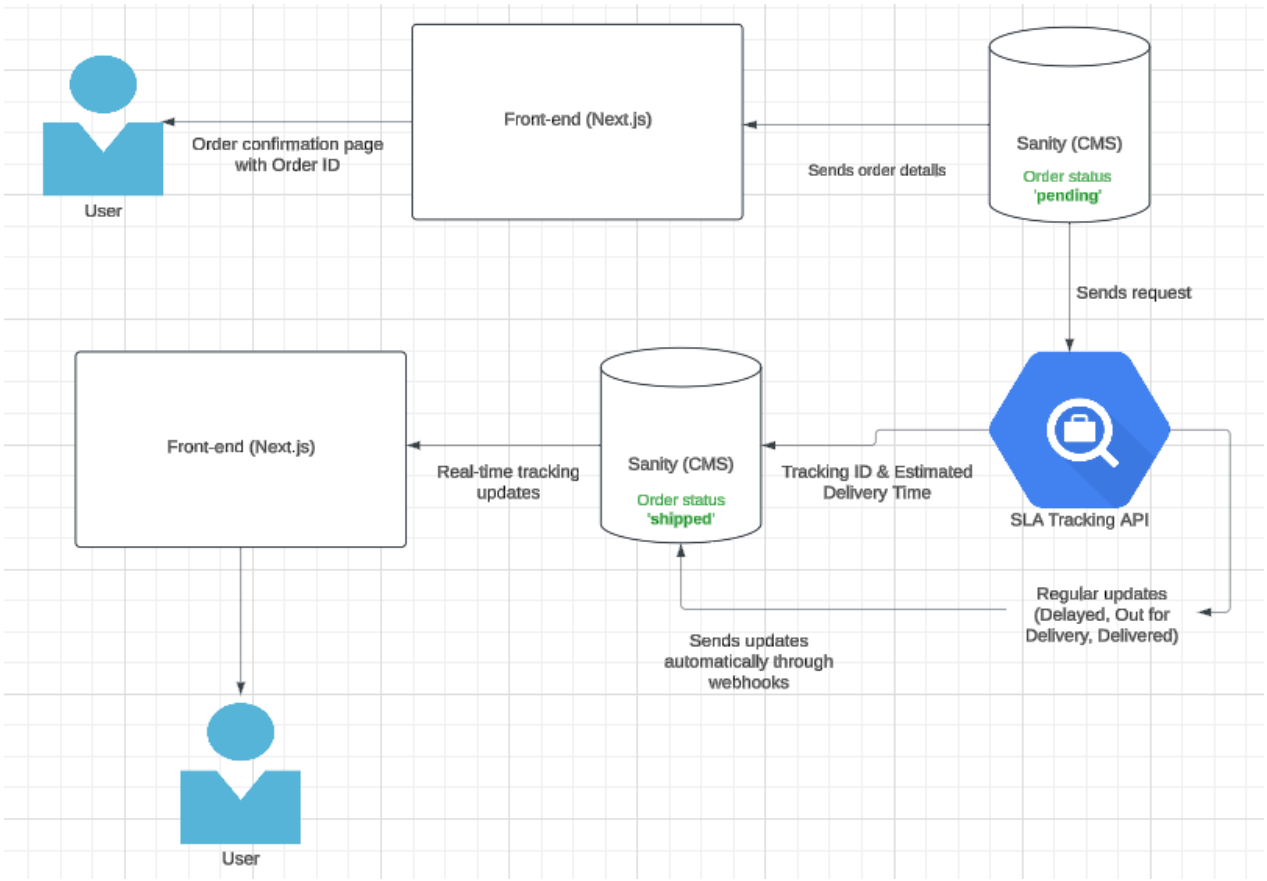
3- User Adds Items to Cart



4- User Proceeds to Checkout & Payment Processing



5- Order Confirmation & Storage & SLA Tracking



Brief Explanation of Each Component

1. **User (Customer)** 👨‍💻
    - Signs up/Signs in, browses products, adds items to the cart, and places an order.
2. **Frontend (Next.js)** 🎨
    - Manages UI and user interactions.
    - Fetches product listings and order details from **Sanity CMS**.
    - Sends payment requests to **Payment API**.
    - Requests order tracking details from **Shipment/SLA Tracking API**.
3. **Sanity CMS (Backend & Database)** 📊
    - Stores product data, customer orders, and user details.
    - Acts as the **main API provider** for fetching and updating orders.
4. **Payment API** 💳
    - Processes online payments securely.
    - Sends success or failure responses to the frontend through CMS.
5. **Shipment API (Tracking)** 🚚
    - Fetches real-time tracking updates for customer orders.
6. **Order Confirmation** ✅
    - Displays confirmation to the user after payment & order placement.
    - Shows tracking details from the **Shipment API**.

API Endpoints

Endpoint Name	Method	Description	Request Example	Response Example
User Sign-Up (/signup)	POST	Registers a new user and stores details in Sanity CMS	{ "name": "Sarah Ali", "email": "sara@example.com", "password": "securepass123" }	{ "userId": 1, "message": "User registered successfully" }
User Login (/signin)	POST	Authenticates the user	{ "email": "sara@example.com", "password": "securepass123" }	{ "userId": 1, "message": "Login successful", "session_id":

				"abc123xyz", "user": { "id": 1, "username": "testuser", "email": "test@example.com" } }
Fetch Menu (/shop)	GET	Retrieves all available items from Sanity CMS.	-	[{ "productId": 101, "name": "Pizza", "price": 12.99, "image": "url_here" }]
Fetch Product Details (/shop/{id})	GET	Retrieves detailed information about a specific product.	-	{ "productId": 101, "name": "Pizza", "price": 12.99, "description": "Delicious Italian pizza", "image": "url_here" }
View Cart (/cart/{userId})	GET	Retrieves the user's cart items.	-	{ "cart": [{ "productId": 101, "name": "Pizza", "quantity": 2 }] }
Checkout (/checkout)	POST	Creates an order and processes payment.	{ "userId": 1, "cart": [{ "productId": 101, "quantity": 2 }], "paymentMethod": "credit_card" }	{ "orderId": 5678, "status": "Processing", "totalAmount": 25.98 }
Fetch Order Status (/order-status/{orderId})	GET	Retrieves the current status of a user's order.	-	{ "orderId": 5678, "status": "Preparing" }
Express Delivery	GET	Fetches	-	{ "orderId":

Status (/express-delivery-status/{orderId})		real-time delivery updates for perishable items.		5678, "status": "In Transit", "ETA": "15 mins" }
Payment Processing (/payment)	POST	Handles payment transactions via a third-party API.	{ "orderId": 5678, "amount": 25.98, "paymentMethod": "credit_card" }	{ "status": "Success", "transactionId": "abc123xyz" }
Payment Webhook (/payment/payment-status)	POST	Listens for payment success/failure updates from the payment API.	{ "transactionId": "abc123xyz", "status": "Success", "orderId": 5678 }	{ "message": "Payment status updated", "orderStatus": "Paid" }
Shipment Tracking (/shipment-tracking/{orderId})		Retrieves shipment updates from a third-party API.	-	{ "orderId": 5678, "status": "Out for Delivery", "ETA": "30 mins" }