System Architecture

1. Frontend (Next.js)

The frontend, developed in **Next.js**, serves as the customer-facing interface where users can browse the menu, place orders, and make payments. The frontend interacts with the following components:

- **Product Data API**: The frontend makes requests to the Sanity CMS to fetch product details such as menu items, prices, descriptions, and images. This allows dynamic loading of the menu.
- **User Interaction**: Customers can view different categories of food (e.g., starters, mains, desserts), select items to add to their cart, customize orders, and proceed to checkout.
- **Checkout**: When a customer proceeds to checkout, the frontend collects the user's information (such as delivery address) and sends it to Sanity CMS and the **Payment Gateway** for processing.

---

2. Sanity CMS

Sanity CMS is the backend content management system that acts as the central database for all the restaurant's data. It stores, manages, and serves the following types of data:

- **Product Data**: Sanity stores detailed information about each menu item, such as names, descriptions, prices, images, and any customization options (e.g., spice level, vegetarian options).
- **Order Records**: After an order is placed, Sanity CMS records the order details, such as the items ordered, customer information, and delivery status.
- **Customer Data**: Sanity stores customer profiles and order history, ensuring personalized experiences for returning customers.

Sanity CMS is crucial for ensuring data is dynamically displayed and updated on the frontend. It also handles storing and managing order data in real-time, which is used for updating customers and staff about the status of their orders.

---

3. Third-Party APIs

Several third-party APIs integrate with the restaurant's website to provide essential services, including payment processing, shipment tracking, and notifications.

Payment Gateway Integration

- **Role**: The **Payment Gateway** (such as Stripe, PayPal, or Square) securely processes customer payments during checkout.
- **Process**:
    - When a customer places an order, the frontend collects the payment details (credit card, digital wallets, etc.).
    - These details are sent to the **Payment Gateway** via secure API calls.
    - Once the payment is successfully processed, a confirmation is returned to the frontend and stored in **Sanity CMS** for future reference.

Shipment Tracking API

- **Role**: The **Shipment Tracking API** (e.g., ShipEngine, Postmates, or local courier APIs) provides real-time updates on the status of customer orders.
- **Process**:
    - When an order is placed and processed, shipment tracking information is sent to the backend, which stores the tracking data.
    - The customer receives a tracking link or status update through the website or an email/SMS notification.
    - The **Frontend** fetches this tracking data from the **Third-Party API** and displays real-time order status updates (e.g., preparing, out for delivery, delivered).

Notification System

- **Role**: A notification service (such as **Twilio** for SMS or **SendGrid** for email) is used to keep the customer informed about the status of their order.
- **Process**:
    - **Order Confirmation**: After payment confirmation, an automated email or SMS is sent to the customer, confirming the order and providing an estimated delivery time.
    - **Order Updates**: Notifications about order status updates (e.g., when the order is out for delivery) are sent via email or SMS.

---

4. Workflow and Data Flow

The following steps outline a typical user journey on your website and how the components work together:

1. User Registration (Optional)

- **Step 1**: A customer registers or logs into the website.
- **Step 2**: The user's data is stored securely in **Sanity CMS**.
- **Step 3**: An email confirmation is sent to the user.

2. Product Browsing

- **Step 1**: A customer browses the menu and selects items to add to their cart.
- **Step 2**: The **Frontend** sends a request to **Sanity CMS** through the **Product Data API**.
- **Step 3**: Sanity CMS responds with product data (name, description, price, image), and the **Frontend** dynamically displays the items.

3. Order Placement

- **Step 1**: The user adds items to their cart and proceeds to checkout.
- **Step 2**: The frontend collects the customer’s details (e.g., delivery address) and order details.
- **Step 3**: This data is sent to **Sanity CMS** for storage and to the **Payment Gateway** for payment processing.
- **Step 4**: The **Payment Gateway** processes the transaction and sends a confirmation back to both the **Frontend** and **Sanity CMS**.

4. Shipment Tracking

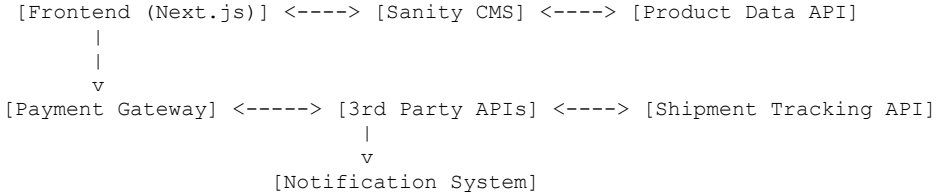
- **Step 1**: Once the order is confirmed, a shipment tracking API request is triggered.
- **Step 2**: The **Third-Party Shipment Tracking API** provides tracking data, which is fetched by the **Frontend**.
- **Step 3**: Customers can see the real-time status of their order (e.g., out for delivery) directly on the website or via a notification.

5. Order Confirmation and Notifications

- **Step 1**: Once payment is confirmed, the **Frontend** sends an order confirmation message to the user.
- **Step 2**: The system sends a **Thank You** email or SMS to the customer, confirming their order and providing an estimated delivery time.

System Diagram Overview

Here’s how the system components interact:



This architecture outlines how different components interact and work together to provide a smooth experience for customers while ensuring efficiency in order processing, payment handling, and delivery tracking. By leveraging **Sanity CMS** for content management, integrating **third-party APIs** for payment and shipment tracking, and offering a responsive **frontend**, the website will deliver an intuitive, functional, and secure service for users and staff alike. This system architecture ensures that all aspects of the restaurant's website — from product browsing to payment processing and order tracking — are seamlessly integrated.