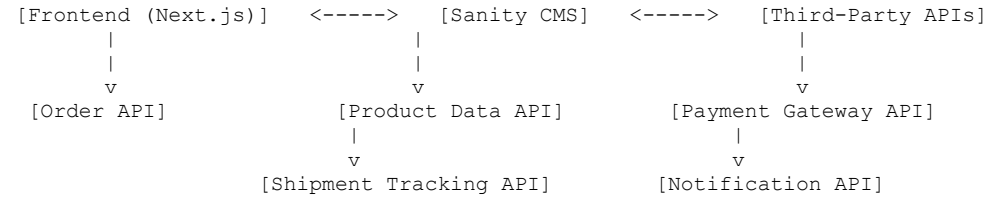
Technical Documentation

1. System Architecture Overview

Architecture Diagram

The architecture of the website involves multiple components working together to deliver a seamless experience for customers and staff. Below is a high-level diagram of how the components interact:



Components and Their Roles

- **Frontend (Next.js)**: The user interface of the website, allowing customers to browse the menu, place orders, and track shipments. It communicates with the backend through APIs to fetch product data, manage orders, and handle payments.
- **Sanity CMS**: The content management system responsible for storing and serving data related to the menu items (products), customer orders, and payment information. Sanity CMS provides a flexible and scalable solution for managing dynamic content and user interactions.
- **Payment Gateway**: Third-party services like Stripe or PayPal handle secure payment processing. The frontend sends payment information, which is securely processed by the payment gateway, and the result is returned to the system.
- **Shipment Tracking API**: An integration with third-party APIs for real-time shipment tracking, providing updates to customers about the status of their orders (e.g., "In Transit," "Out for Delivery").
- **Notification API**: This service sends notifications to customers via email or SMS for order confirmation, shipping updates, and delivery status.

---

2. Key Workflows

User Browses the Menu

1. The user visits the website and selects a category (e.g., Appetizers, Main Courses, Desserts).
2. The frontend sends a GET request to the **Product Data API** to fetch menu items.
3. The API returns a list of products, including names, descriptions, prices, and images, which are displayed on the website.

User Adds Products to Cart

1. The user selects items to add to their cart.
2. The frontend sends a POST request to the **Order API** with the selected products and quantities.
3. The **Order API** returns a confirmation with the order details and total price.

User Places an Order

1. The user proceeds to checkout, providing their delivery details and payment information.
2. The frontend sends the order details to the **Sanity CMS** and **Payment Gateway API**.
3. The **Payment Gateway** processes the payment, and upon success, a confirmation is sent to both the frontend and **Sanity CMS**.
4. The user receives an order confirmation via email or SMS from the **Notification API**.

Order Shipment Tracking

1. After the order is placed, the **Shipment Tracking API** fetches real-time updates on the delivery status.
2. The frontend checks the API for status updates (e.g., "Out for Delivery") and shows the information to the customer.
3. The customer can track the delivery in real-time through the website.

3. Category-Specific Instructions

Q-Commerce Workflow

- **Express Delivery**: For customers placing urgent orders (e.g., last-minute lunch or dinner orders), the system fetches real-time delivery updates using the **/express-delivery-status** endpoint.

  Example Endpoint:

  - Endpoint: `/express-delivery-status`
  - Method: `GET`
  - Description: Fetch real-time delivery status for urgent orders.
  - Response Example:

```
{
  "orderId": 123,
  "status": "In Transit",
  "ETA": "15 mins"
}
```

General eCommerce Workflow

- **Product Browsing**: The user browses the menu and selects items. The system fetches the product details from Sanity CMS via the **/products** endpoint.

  Example Endpoint:

  - Endpoint: `/products`
  - Method: `GET`
  - Description: Fetch all product details (menu items) for browsing.
  - Response Example:

```
[
 {
   "id": 1,
   "name": "Spaghetti Carbonara",
   "price": 15.99,
   "stock": 50,
   "image": "https://example.com/spaghetti-carbonara.jpg"
 },
 {
   "id": 2,
   "name": "Margherita Pizza",
   "price": 12.99,
   "stock": 30,
   "image": "https://example.com/margherita-pizza.jpg"
 }
]
```

4. API Endpoints

Endpoint	Method	Purpose	Response Example
`/products`	`GET`	Fetch all product details	`{ "id": 1, "name": "Spaghetti Carbonara", "price": 15.99 }`
`/orders`	`POST`	Create a new order	`{ "orderId": 101, "status": "Success", "message": "Order placed successfully" }`
`/order/{id}`	`GET`	Fetch details of a specific order	`{ "orderId": 101, "status": "Processing", "totalPrice": 44.97 }`

Endpoint	Method	Purpose	Response Example
/shipment/{orderId}	GET	Track shipment status	{ "orderId": 101, "status": "In Transit", "ETA": "15 mins" }
/payment	POST	Process payment	{ "paymentId": 5001, "status": "Success", "message": "Payment processed" }
/send-notification	POST	Send order notifications (email/SMS)	{ "status": "Success", "message": "Notification sent successfully" }

---

5. Sanity Schema Example

Here's an example of a data schema for the **Product** document in Sanity CMS:

```
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: 'description',
      type: 'text',
      title: 'Description'
    },
    {
      name: 'image',
      type: 'image',
      title: 'Product Image'
    }
  ]
};
```

This schema allows easy management of products in Sanity, including the name, price, stock, description, and image for each menu item.

---

6. Technical Roadmap

The following is a general technical roadmap for building and launching the restaurant website:

- **Phase 1: Requirements Gathering and System Design** (Weeks 1-2)
  - Finalize business requirements and features.
  - Design the system architecture and workflows.
  - Define data schema and API endpoints.
- **Phase 2: Development and Integration** (Weeks 3-6)
  - Develop the frontend using Next.js.
  - Set up Sanity CMS for product and order management.
  - Integrate third-party APIs for payment, shipment tracking, and notifications.
- **Phase 3: Testing and Quality Assurance** (Weeks 7-8)
  - Perform unit tests and integration tests for all APIs.

    - Conduct usability and performance testing for the frontend.
- **Phase 4: Launch and Maintenance** (Week 9 and onwards)
    - Launch the website.
    - Monitor the website for issues and provide ongoing support.

Conclusion

This technical documentation outlines the system architecture, workflows, API requirements, and data schema necessary for building the restaurant's website. It serves as a blueprint for development and ensures a streamlined process for implementing the various components that interact with the frontend, backend, and third-party services.