PLANNING THE TECHNICAL FOUNDATION

Q-Commerce

FOODTUCK RESTARUANT

Technical Requirements:

Frontend (Next.js)

- Your frontend is built with Next.js, which interacts with several APIs for dynamic content and functionalities.

Checkout Page

- The checkout page sends requests to the Checkout API to process the order and interact with the **Order Data API**, which stores and manages order-related information in the **Database**.

Product Page

- The product page fetches product data from the **Product API**, which retrieves product information from **Sanity CMS** for dynamic product content.

Menu Page

- The menu page fetches data about different food items using the **Menu API**, which interacts with a **Food Items Data API** to get menu data from the **Database**.

Chef Page

- The chef page interacts with the **Chef API**, which fetches chef-related data from the **Chef Data API** stored in the **Database**.

Third-Party API

- A third-party API is used for external integrations, such as shipping or product tracking. It communicates with the **Shipment Tracking API**, which connects to the **Shipping Provider** for tracking orders.

Payment Gateway

- The **Payment Gateway** processes payments through an external **Payment Provider** using a **Payment API**.

System Architecture

[Frontend (Next.js)]

|

|-----> [Checkout Page] <-----> [Checkout API] <-----> [Order Data API] <-----> [Database]

| |

| |-----> [Product Page] <-----> [Product API] <-----> [Product Data API] <-----> [Sanity CMS]

| |

| |-----> [Menu Page] <-----> [Menu API] <-----> [Food Items Data API] <-----> [Database]

| |

| |-----> [Chef Page] <-----> [Chef API] <-----> [Chef Data API] <-----> [Database]

```
   |
   |-----> [Third-Party API] -----> [Shipment Tracking API] <-----> [Shipping API] <-
----> [Shipping Provider]
   |
   |-----> [Payment Gateway] <-----> [Payment API] <-----> [Payment Provider]
```

Key Workflows:

1. **User Registration:**
   - **Step 1**: User signs up through the frontend (Next.js).
   - **Step 2**: The registration form data is sent to an API endpoint.
   - **Step 3**: The API processes the data and stores the user's information in **Sanity CMS**.
   - **Step 4**: A confirmation email is sent to the user via a third-party service (e.g., email API).

1. **Product Browsing:**
   - **Step 1**: User visits the product page on the frontend.
   - **Step 2**: The frontend makes an API request to the **Product API**.
   - **Step 3**: The **Product API** retrieves data from **Sanity CMS**.
   - **Step 4**: Product data (e.g., name, image, price) is returned to the frontend and displayed to the user.

1. **Menu Browsing:**
   - **Step 1**: User navigates to the food menu page.
   - **Step 2**: The frontend makes an API request to the **Menu API**.
   - **Step 3**: The **Menu API** fetches food items from the **Food Items Data API** stored in the **Database**.
   - **Step 4**: Food items (e.g., name, description, calories, price) are returned to the frontend and displayed in a tabbed layout for breakfast, lunch, dinner, etc.

1. **Chef Browsing:**
   - **Step 1**: User visits the chef section.
   - **Step 2**: Frontend requests chef data through the **Chef API**.
   - **Step 3**: The **Chef API** fetches chef details from the **Chef Data API** stored in the **Database**.

    - **Step 4**: Chef information (e.g., name, image) is displayed on the frontend.

1. **Order Placement:**
    - **Step 1**: User adds items to the cart.
    - **Step 2**: User proceeds to the checkout page.
    - **Step 3**: The frontend collects order details (items, shipping info, etc.).
    - **Step 4**: Order details are sent to the **Checkout API**.
    - **Step 5**: The **Checkout API** processes the order and saves it to the **Order Data API** in the **Database**.
    - **Step 6**: A confirmation is sent to the user (via email or SMS) with order details.

1. **Shipment Tracking:**
    - **Step 1**: User requests order status update.
    - **Step 2**: The frontend requests order tracking information from a **Third-Party API**.
    - **Step 3**: The **Third-Party API** queries the **Shipment Tracking API**, which gets real-time data from the **Shipping Provider**.
    - **Step 4**: The updated shipment status is returned to the frontend and displayed to the user (e.g., "Out for Delivery", "Shipped", etc.).

1. **Payment Processing:**
    - **Step 1**: User proceeds to payment after confirming the order.
    - **Step 2**: The frontend sends payment information (amount, payment method, etc.) to the **Payment API**.
    - **Step 3**: The **Payment API** interacts with the **Payment Gateway** to process the transaction.
    - **Step 4**: Once the payment is processed, the **Payment API** returns a success or failure message.
    - **Step 5**: A confirmation (success or failure) is displayed to the user on the frontend, and the order status is updated in the **Order Data API**.

1. User Registration and Authentication APIs:

Endpoint 1: /register

Method: POST

Description: Allows new users to register by submitting their personal details (name, email, password).

Request Example:

```
{
  "name": "John Doe",
  "email": "john.doe@example.com",
  "password": "securePassword123"
}
```

Response Example:

```
{
  "message": "User registered successfully",
  "userId": 12345,
  "token": "jwt_token_here"
}
```

<u>Endpoint 2: /login</u>

Method: POST

Description: Authenticates an existing user based on provided email and password.

Request Example:

```
{
  "email": "john.doe@example.com",
  "password": "securePassword123"
}
```

Response Example:

```
{
  "message": "Login successful",
  "userId": 12345,
  "token": "jwt_token_here"
}
```

2. Product and Menu APIs:

Endpoint 1: /products

Method: GET

Description: Fetches a list of all products available for browsing.

Response Example:

```
[
  {
    "id": 1,
    "name": "Americano",
    "description": "Freshly brewed Americano",
    "price": 5.99,
    "imageUrl": "/images/americano.jpg"
  },
  {
    "id": 2,
    "name": "Espresso",
    "description": "Rich and bold espresso",
    "price": 3.99,
    "imageUrl": "/images/espresso.jpg"
  }
]
```

Endpoint 2: /menu

Method: GET

Description: Retrieves the menu items categorized by meal type (e.g., Breakfast, Lunch, etc.).

Response Example:

```
{
  "breakfast": [
    {
      "id": 1,
      "name": "Pancakes",
      "description": "Fluffy pancakes with syrup",
      "calories": 350,
      "price": 7.99
    }
  ],
  "lunch": [
    {
      "id": 2,
      "name": "Caesar Salad",
      "description": "Crisp lettuce with creamy dressing",
      "calories": 250,
      "price": 8.99
    }
  ]
}
```

3. Order Placement and Cart APIs:

<u>Endpoint 1: /add-to-cart</u>

Method: POST

Description: Adds a product to the user's cart.

Request Example:

```
{
  "userId": 12345,
  "productId": 1,
  "quantity": 2
}
```

Response Example:

```
{
  "message": "Product added to cart successfully",
  "cartId": 6789
}
```

Endpoint 2: /checkout

Method: POST

Description: Initiates the checkout process, saving order details and calculating total.

Request Example:

```
{
  "userId": 12345,
  "cartId": 6789,
  "paymentMethod": "credit_card",
  "address": "123 Main St, Anytown, USA"
}
```

Response Example:

```
{
  "orderId": 123,
  "totalAmount": 29.99,
  "status": "Order placed successfully"
}
```

4. Payment API:

Endpoint 1: /payment

Method: POST

Description: Processes the payment for the order.

Request Example:

```
{
  "orderId": 123,
  "paymentDetails": {
    "cardNumber": "4111111111111111",
    "expiryDate": "12/23",
    "cvv": "123"
  }
}
```

Response Example:

```
{
  "message": "Payment successful",
  "transactionId": "txn_123456",
  "status": "Paid"
}
```

5. Shipment Tracking API:

Endpoint 1: /shipment-status

Method: GET

Description: Fetches real-time shipment status updates for an order.

Request Example:

```
{
  "orderId": 123
}
```

Response Example:

```
{
  "orderId": 123,
  "status": "Shipped",
  "ETA": "3 hours",
  "trackingNumber": "TRACK12345"
}
```

Endpoint 2: /shipment-tracking

Method: GET

Description: Retrieves the detailed tracking information for the shipment.

Request Example:

```
{
  "trackingNumber": "TRACK12345"
}
```

Response Example:

```
{
  "trackingNumber": "TRACK12345",
  "status": "In Transit",
  "location": "Warehouse in Anytown",
  "ETA": "2 hours"
```

}

6. Customer Feedback API:

Endpoint 1: /submit-feedback

Method: POST

Description: Allows users to submit feedback on their order or experience.

Request Example:

```
{
  "userId": 12345,
  "orderId": 123,
  "rating": 5,
  "comment": "Great food and quick delivery!"
}
```

Response Example:

```
{
  "message": "Feedback submitted successfully",
  "status": "Received"
}
```

7. Product Recommendations API:

Endpoint 1: /recommendations

Method: GET

Description: Fetches personalized product recommendations based on user preferences or browsing history.

Request Example:

```
{
  "userId": 12345
}
```

Response Example:

```
[
  {
    "id": 1,
    "name": "Latte",
    "description": "Smooth and creamy",
    "price": 4.99,
    "imageUrl": "/images/latte.jpg"
  },
  {
    "id": 2,
    "name": "Cappuccino",
    "description": "Foamy and rich",
    "price": 5.49,
    "imageUrl": "/images/cappuccino.jpg"
  }
]
```

Product Schema:

```
// schemas/product.ts
export default {
  name: 'product',
  type: 'document',
  title: 'Product',
 fields: [
  { name: 'name', type: 'string', title: 'Product Name' },
  { name: 'description', type: 'text', title: 'Description' },
  { name: 'price', type: 'number', title: 'Price' },
  { name: 'stock', type: 'number', title: 'Stock Level' },
  { name: 'image', type: 'image', title: 'Image' },
  { name: 'category', type: 'string', title: 'Category' },
  { name: 'rating', type: 'number', title: 'Rating' }
 ]
};
```

Example API Endpoints Based on Sanity CMS Schema

1. **Get Products**
   - **Endpoint: /api/products**
   - **Method: GET**
   - **Description: Fetches a list of all products.**
   - **Response Example:**
   - **[**
   - **{**
   - **"id": 1,**
   - **"name": "Americano",**
   - **"price": 5.99,**
   - **"stock": 10,**
   - **"image": "/images/americano.jpg",**
   - **"rating": 4.5,**
   - **"category": "Coffee"**
   - **},**
   - **{**
   - **"id": 2,**
   - **"name": "Espresso",**
   - **"price": 3.99,**
   - **"stock": 15,**

-     "image": "/images/espresso.jpg",
-     "rating": 4.7,
-     "category": "Coffee"
-   }
-]

1. **Get Order Details**
   - **Endpoint: /api/orders/{orderId}**
   - **Method: GET**
   - **Description: Retrieves details of a specific order by its ID.**
   - **Response Example:**
   - **{**
   - **"orderId": 123,**
   - **"user": "John Doe",**
   - **"products": [**
   - **{**
   - **"name": "Americano",**
   - **"quantity": 2,**
   - **"price": 5.99**
   - **}**
   - **],**
   - **"totalAmount": 11.98,**
   - **"status": "Shipped",**
   - **"paymentStatus": "Paid",**
   - **"shipmentStatus": "On the way",**
   - **"address": "123 Main St, Anytown, USA"**
   - **}**

1. **Place Order**
   - **Endpoint: /api/place-order**
   - **Method: POST**
   - **Description: Creates a new order after user checkout.**
   - **Request Example:**
   - **{**
   - **"userId": 123,**
   - **"products": [**
   - **{ "productId": 1, "quantity": 2 },**

- { "productId": 2, "quantity": 1 }
-],
- "totalAmount": 17.97,
- "shippingAddress": "123 Main St, Anytown, USA"
- }
- Response Example:
- {
- "message": "Order placed successfully",
- "orderId": 124
- }