**Q-Commerce Food Marketplace – Technical Documentation**

**Overview**

This platform is designed to revolutionize ultra-fast food delivery services, ensuring seamless order processing, real-time tracking, and efficient logistics. The system integrates a modern frontend, a robust content management backend, and multiple third-party APIs to create a high-performance, scalable, and responsive experience for users.

**Frontend – Next.js**

The user interface is built with **Next.js**, a powerful React framework that enhances performance through server-side rendering (SSR), static site generation (SSG), and optimized routing.

**Key Features**

* **Server-Side Rendering (SSR):** Improves SEO and accelerates initial load times.
* **Dynamic Routing:** Enables seamless navigation between pages.
* **API Routes:** Manages server-side logic efficiently.
* **Image Optimization:** Uses Next.js Image component for faster loading and better performance.
* **Responsive Design:** Built with Tailwind CSS for an adaptive and mobile-friendly interface.

**Core Components**

* **Home Page:** Showcases featured restaurants, trending dishes, and personalized recommendations.
* **Restaurant Listings:** Displays eateries in grid or list views with advanced filtering and sorting options.
* **Menu Page:** Presents detailed menu items with customization features.
* **Cart & Checkout:** Manages order placement, item selection, and secure payments.
* **Order Tracking:** Provides real-time updates on order status and estimated delivery time.
* **User Profile:** Allows users to manage personal details, saved addresses, and order history

**Backend – Sanity CMS**

A dynamic and flexible content management system powers the backend, enabling real-time updates and seamless data handling.

**Key Features**

* **Custom Schemas:** Structured data models for restaurants, menus, orders, and user profiles.

**. Instant Content Updates:** Ensures real-time synchronization across the platform.

* **API for Data Management:** Efficiently retrieves and manipulates stored information.
* **Content Versioning & Rollback:** Allows restoring previous content versions when needed.

**WebSocket Server**

A dedicated WebSocket server ensures instant communication between the frontend and backend, enabling real-time updates.

**Key Features**

* **Live Order Status Updates:** Keeps customers informed about their order progress.
* **Real-Time Delivery Tracking:** Ensures accurate monitoring of food deliveries.
* **Dynamic Inventory Management:** Automatically reflects menu item availability.

**Third-Party API Integrations**

**1. Auth0 (Authentication & Authorization)**

* Ensures secure user authentication and access control.
* Supports social media logins for a seamless experience.
* Implements token-based authentication for enhanced security.

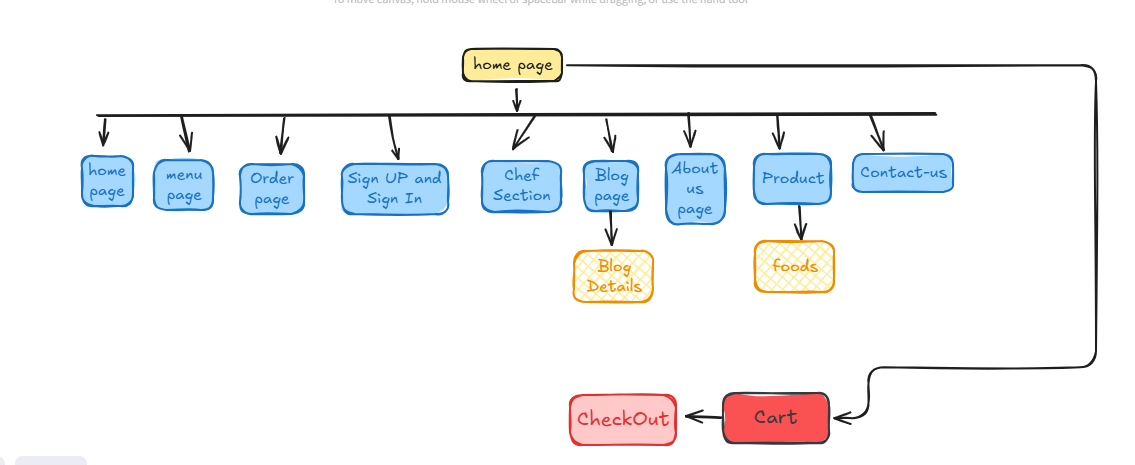
**2. Stripe API (Payment Processing)**

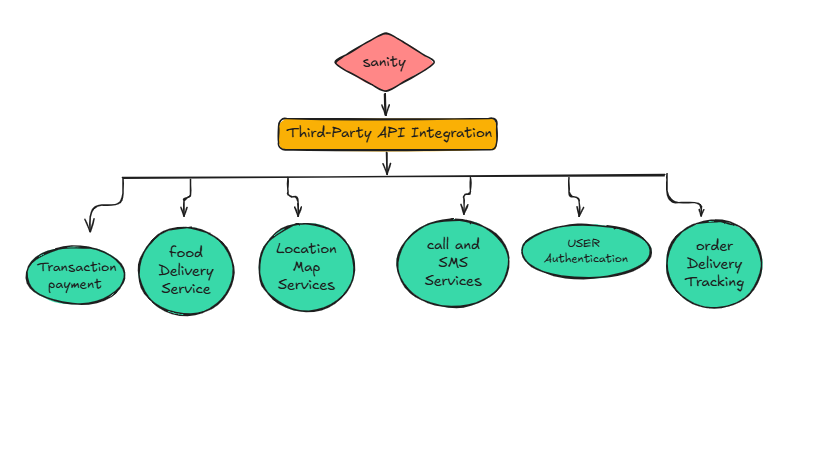
* Facilitates secure transactions and payment processing.
* Manages refunds and dispute resolutions.
* Supports multiple payment methods, including credit cards, wallets, and UPI.

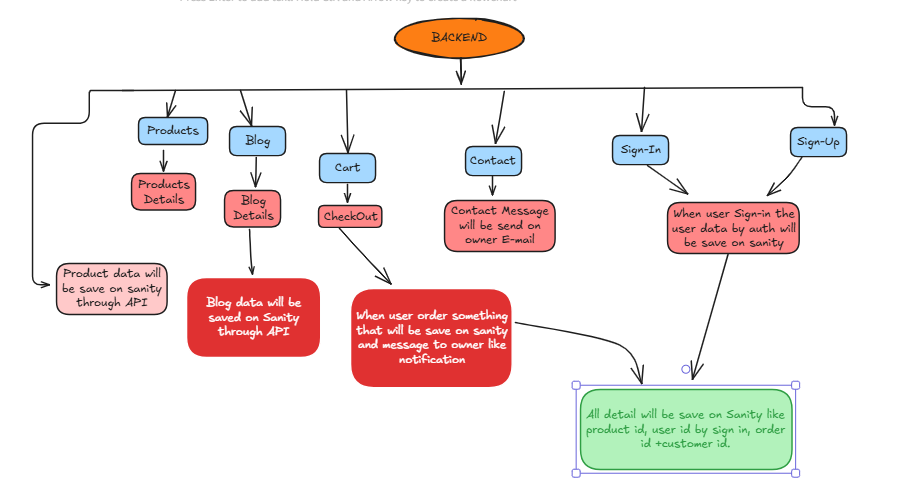
**3. Google Maps API (Location & Tracking)**

* Provides precise restaurant location data.Enables real-time tracking of delivery personnel.

System Architecture







**Key Workflows**

|  |
| --- |
| **1. User Registration** |
| * User signs up with name, email, and password. |
| * Data stored in **Sanity CMS**. |
| * Verification email/SMS sent. |
| * After verification, user can log in. |
| **2. Product Browsing** |
| * User views food categories. |
| * Frontend fetches product data from **Sanity API**. |
| * Products displayed with filters (price, category, restaurant). |
| **3. Order Placement** |
| * User adds items to the cart. |
| * Proceeds to checkout and enters details. |
| * Payment processed via **third-party API**. |
| * Order saved in **Sanity CMS**, confirmation sent. |
| **4. Shipment Tracking** |
| * Order status updated by restaurant. |
| * Real-time tracking via **third-party API**. |
| * User can view delivery status & estimated time. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Endpoint** | **Method** | **Purpose** | **Response Example** | | --- | --- | --- | --- | | /api/auth/signup | POST | User  registration | { "message": "User registered successfully", "userId": "1234abcd" } | | /api/auth/login | POST | User login | { "token": "jwt-token", "userId": "1234abcd" } | | /api/restaurants | GET | Fetches all restaurants | [ { "id": "restaurant1", "name": "Pizza Place", "location": { "lat": 40.7128, "lng": -74.0060 } }, { "id": "restaurant2", "name": "Burger Joint", "location": { "lat": 40.730610, "lng": -73.935242 } } ] | | /api/restaurants/{id} | GET | Fetch details of a specific restaurant | { "id": "restaurant1", "name": "Pizza Place", "description": "Best pizza in town!", "menu": [ { "id": "pizza1", "name": "Margherita", "price": 9.99 }, { "id": "pizza2", "name": "Pepperoni", "price": 12.99 } ] } | | /api/menu | GET | Fetch all menu items | [ { "id": "pizza1", "name": "Margherita", "price": 9.99, "category": "Pizza", "availability": true }, { "id": "burger1", "name": "Cheeseburger", "price": 7.99, "category": "Burger", "availability": true } ] | | /api/menu/{id} | GET | Fetch a specific menu item | { "id": "pizza1", "name": "Margherita", "description": "Classic Margherita with mozzarella and basil", "price": 9.99, "category": "Pizza", "availability": true } | | /api/orders | POST | Place a new order | { "orderId": "order123", "status": "pending", "totalPrice": 29.97, "message": "Order placed successfully" } | | /api/orders/{id} | GET | Fetch a specific order | { "orderId": "order123", "userId": "1234abcd", "items": [ { "menuItemId": "pizza1", "quantity": 2 }, { "menuItemId": "burger1", "quantity": 1 } ], "totalPrice": 29.97, "status": "shipped" } | | /api/tracking/{orderId} | GET | Track order status | { "orderId": "order123", "status": "shipped", "estimatedDeliveryTime": "12:30 PM" } | | /api/payment | POST | Process a payment | { "paymentStatus": "success", "transactionId": "txn123456" } | |