

Food Website Documentation

Prepared By: Rabia Sohail

1. Business Overview

1.1 Purpose

The purpose of this food website is to create an online platform where customers can:

- Browse a variety of food items, meals, or beverages.
- Place online orders for delivery or pickup.
- Track their orders in real-time.

1.2 Goals

- Provide a seamless food ordering experience.
- Ensure accurate and timely delivery of orders.
- Build customer loyalty through excellent service and user experience.

1.3 Target Audience

- Individuals aged 18-45.
- Office workers, students, and busy professionals looking for convenient food options.
- Food enthusiasts who want high-quality meals delivered to their doorstep.

1.4 Unique Selling Points

- Real-time order tracking.
 - Curated menu with locally sourced ingredients.
 - Integration with top-notch logistics providers for fast delivery.
 - User-friendly interface with personalized recommendations.
-

2. Features and Functionality

2.1 User Features

1. **Authentication**
 - Users can sign up, log in, and log out using [Clerk](#).
 - Social login options (Google, Facebook, etc.).
2. **Food Menu**
 - View categorized food items (e.g., meals, beverages, desserts).

- Search and filter options (e.g., by cuisine, price, or dietary preferences).
- 3. **Ordering System**
 - Add items to the cart.
 - Modify quantities before checkout.
 - Apply promo codes and discounts.
- 4. **Payment Gateway**
 - Integration with payment providers (Stripe).
- 5. **Order Tracking**
 - Real-time order tracking using [ShipEngine](#).
 - Notifications for order status (e.g., order confirmed, out for delivery).
- 6. **Reviews and Ratings**
 - Users can rate their food and delivery experience.
 - Option to leave feedback for continuous improvement.

2.2 Admin Features

1. **Dashboard**
 - Manage menu items (add, update, delete).
 - View and manage user accounts.
 - Monitor order statuses.
 2. **Order Management**
 - View incoming orders in real time.
 - Assign orders to delivery personnel.
 3. **Analytics and Reporting**
 - Sales data visualization.
 - Customer feedback analysis.
 - Track popular menu items.
-

3. Technical Overview

3.1 Tech Stack

- **Frontend:** React with Next.js and Tailwind CSS for styling.
- **Backend:** Next.js API routes and Sanity for content management.
- **Database:** Sanity for storing product details and user data.
- **Authentication:** Clerk for user management and secure login.
- **Order Tracking:** ShipEngine for real-time delivery updates.

3.2 APIs

1. **Sanity CMS**
 - Manages food items, categories, and blog content.
 - Fetch data using GROQ queries.

2. **Clerk**
 - Handles user authentication, profile management, and session handling.
3. **ShipEngine**
 - Provides shipping rates, label creation, and order tracking.
4. **Stripe**
 - Facilitates secure online payments.

3.3 Database Structure

- **User**
 - ID, Name, Email, Password, Address, Order History.
 - **Food Items**
 - ID, Name, Category, Price, Ingredients, Image URL.
 - **Orders**
 - Order ID, User ID, Food Items, Total Amount, Status.
 - **Delivery**
 - Order ID, Tracking Number, Status, Estimated Delivery Time.
-

4. Workflow

4.1 User Workflow

1. **Sign-Up/Login:** Users register or log in using Clerk.
2. **Browse Menu:** Explore food items from the categorized menu.
3. **Place Order:** Add items to the cart, confirm details, and make payment.
4. **Order Confirmation:** Receive confirmation via email or notification.
5. **Track Order:** Real-time updates via ShipEngine.
6. **Delivery:** Receive food and leave feedback.

4.2 Admin Workflow

1. **Menu Management:** Add or update food items in Sanity.
2. **Order Processing:** Approve and assign delivery using the admin dashboard.
3. **Delivery Coordination:** Use ShipEngine to track and manage deliveries.
4. **Analytics Monitoring:** Review performance metrics and customer feedback.

WorkFlow Diagram

[Frontend (Next.js)]

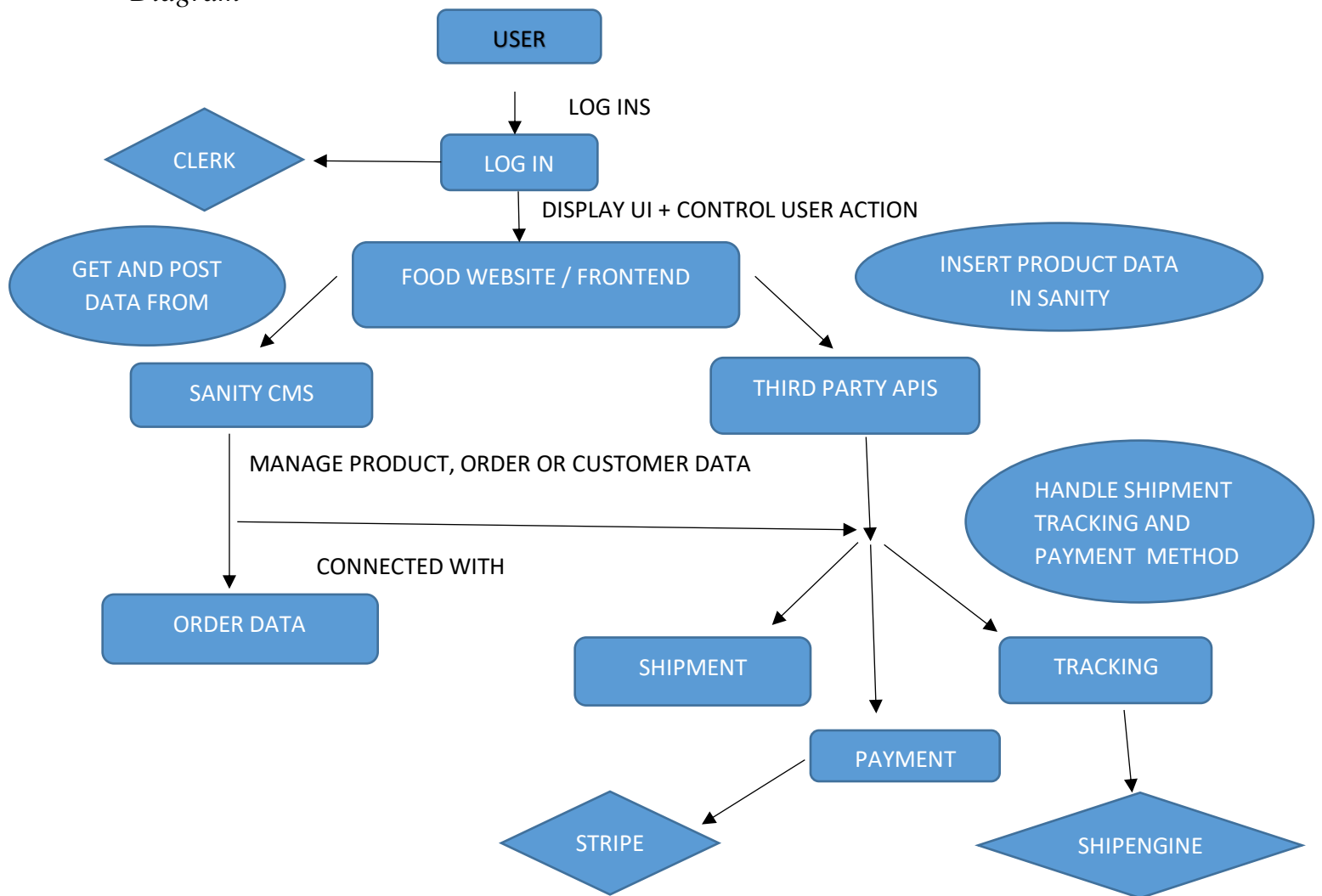
|

```

[Sanity CMS] <-----> [Product Data API]
|
[Third-Party APIs] ---> [Shipment Tracking API]
|
                    ---> [Payment Gateway]

```

Diagram



5. API Integration Steps

5.1 Sanity CMS

1. Create schemas for Food Items, Categories, and Orders.
2. Use GROQ queries to fetch data for the menu and blog

Schemas

Food Schema

```
export default {
  name: 'food',
  type: 'document',
  title: 'Food',
  fields: [
    {
      name: 'title',
      type: 'string',
      title: 'Food Title',
    },
    {
      name: 'category',
      type: 'string',
      title: 'Category',
      description:
        'Category of the food item (e.g., Burger, Sandwich, Drink, etc.)',
    },
    {
      name: 'price',
      type: 'number',
      title: 'Current Price',
    },
    {
      name: 'rating',
      type: 'number',
      title: 'Rating',
      description: 'Customers rating',
    },
    {
      name: 'image',
      type: 'image',
      title: 'Food Image',
    },
  ],
}
```

```

    options: {
      hotspot: true,
    },
  },
  {
    name: 'description',
    type: 'text',
    title: 'Description',
    description: 'Short description of the food item',
  },
  {
    name: "id",
    type: "string",
    title: "Id"
  },
  {
    name: 'stock',
    type: 'boolean',
    title: 'Available',
    description: 'Availability status of the food item',
  },
],
};

```

Chef Schema

```

export default {
  name: 'chef',
  type: 'document',
  title: 'Chef',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Chef Name',
    },
    {
      name: 'position',
      type: 'string',
      title: 'Position',
      description: 'Role or title of the chef (e.g., Head Chef, Sous Chef)',
    },
  ],
};

```

```

    name: 'experience',
    type: 'number',
    title: 'Years of Experience',
    description: 'Number of years the chef has worked in the culinary field',
  },
  {
    name: 'specialty',
    type: 'string',
    title: 'Specialty',
    description: 'Specialization of the chef (e.g., Italian Cuisine,
Pastry)',
  },
  {
    name: 'image',
    type: 'image',
    title: 'Chef Image',
    options: {
      hotspot: true,
    },
  },
  {
    name: 'description',
    type: 'text',
    title: 'Description',
    description: 'Short bio or introduction about the chef',
  },
  {
    name: 'available',
    type: 'boolean',
    title: 'Currently Active',
    description: 'Availability status of the chef',
  },
],
};

```

Order Schema

```

export default {
  name: 'order',
  type: 'document',
  fields: [
    { name: 'customer',
      type: 'reference',

```

```

      to: [
        { type: 'customer' }],
      title: 'Customer'
    },
    { name: 'products',
      type: 'array',
      of: [
        { type: 'reference',
          to: [
            { type: 'product' }
          ]
        }
      ]
    }
  ]],
  title: 'Products' },

  { name: 'totalAmount',
    type: 'number',
    title: 'Total Amount' },
  {
    name: 'status',
    type: 'string',
    title: 'Order Status' }
]
};

```

GROQ Queries

Food

```

const query = `*[_type == "food"]{
  _id,
  title,
  price,
  "imageUrl": image.asset->url,
  description,
  rating,
  review,
  stock
}`;
const data: Product[] = await client.fetch(query);

```


Chef

```
const query = `*[_type=="chef"]{
  _id,
  name,
  specialty,
  "imageUrl":image.asset->url,
}`
const chef = await client.fetch(query)
```

5.2 Clerk Authentication

1. Install Clerk SDK.
2. Set up authentication routes for sign-up, login, and logout.
3. Restrict access to certain pages using Clerk middleware.

5.3 ShipEngine

1. Create a ShipEngine account.
2. Use API keys to integrate real-time order tracking.
3. Fetch delivery status and display it on the user's dashboard.

5.4 Stripe Payment

1. Create a Stripe account.
2. Set up payment intents for secure transactions.
3. Confirm payment status before order confirmation.