# Market Blace Builder Hackathon 2025 (Day 2)

# Marketplace Technical Foundation

# "Foodtuck: A Q-Commerce Marketplace"

## 1. Define Technical Requirements:

### Frontend Requirements:

1) **User-Friendly Interface:**
   - Create a clean and attractive user interface for browsing food items from restaurants.
2) **Responsive Design:**
   - Ensure compatibility across mobile and desktop devices with a responsive layout.
3) **Essential Pages:**
   - **Home Page:** Displays featured food items and categories.
   - **Product Listing:** Showcases menu items based on category or restaurant.
   - **Product Details**: Displays details like ingredients, price, and availability.
   - **Cart:** Summarizes selected items for checkout.
   - **Checkout & Order Confirmation:** Handles the payment and shows order confirmation.

### Sanity CMS as Backend:

1) Use **Sanity CMS** to manage:
   - Food product data.
   - Customer information.
   - Order details.
2) Design schemas in Sanity CMS for:
   - **Products:** Includes title, description, price, image, and availability status.
   - **Orders:** Tracks customer details, product IDs, quantities, and timestamps.
   - **Customers:** Includes name, contact info, and address.

### Third-Party APIs:

1) Integrate APIs for:
   - Shipment tracking.
   - Payment processing.

- Real-time notifications (e.g., Twilio for SMS or email confirmations).

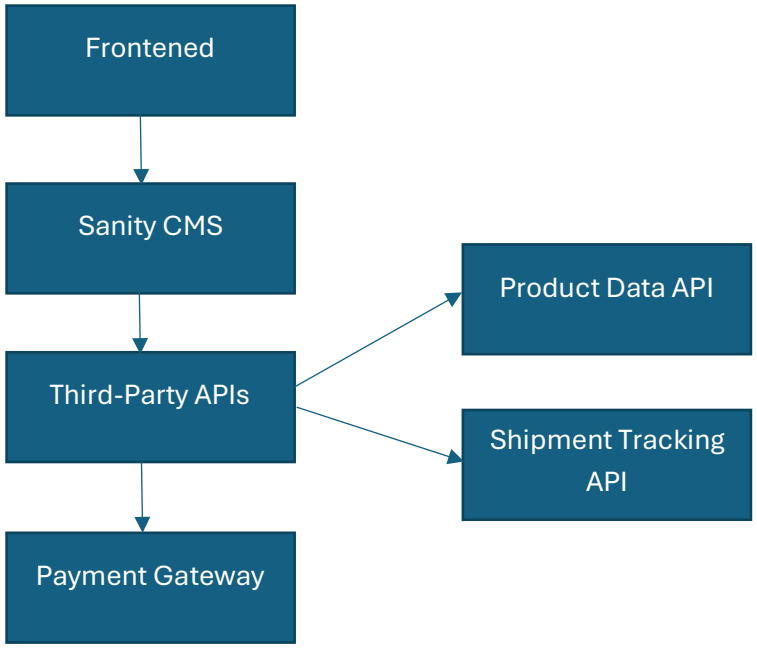
2) Ensure APIs are well-documented and provide necessary data to the frontend.

# 2. Design System Architecture

## System Components:

1) **Frontend (Next.js):**
   - Manages UI interactions and sends requests to the backend.
2) **Sanity CMS:**
   - Acts as the database for food items, orders, and customer data.
3) **Third-Party APIs:**
   - Handles shipment tracking, payment gateways, and notifications.

### System Architecture Diagram




## Architecture Workflow:

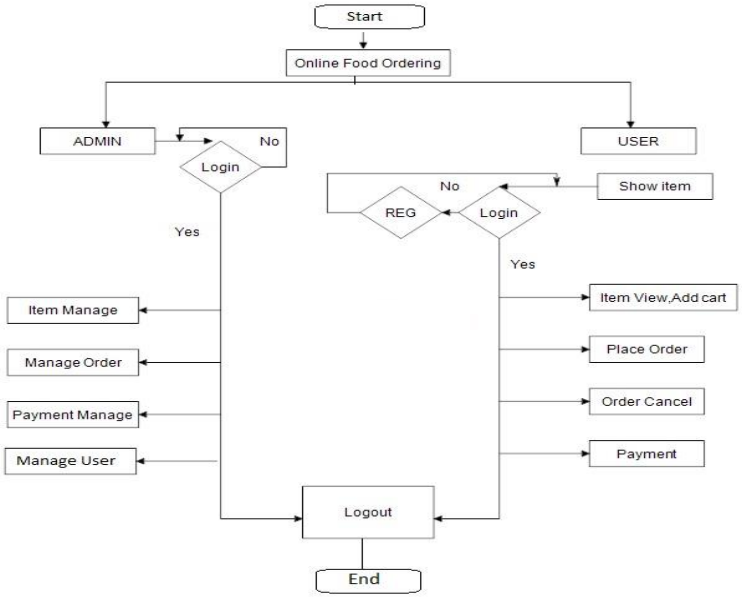
1. A user browses food items on the marketplace (Next.js frontend).
2. The frontend requests the Product Data API (via Sanity CMS) to fetch items dynamically.
3. When an order is placed:
   - The order details are sent to Sanity CMS via API.

- Shipment tracking updates are fetched using a Third-Party API and displayed.
- Payment processing is handled via the Payment Gateway, with confirmation recorded in Sanity CMS.

# 3. Key Workflows:

1) **User Registration:**
   - User signs up -> Data is stored in Sanity -> Confirmation sent to the user.
2) **Product Browsing:**
   - User views food categories -> Sanity API fetches data -> Items displayed.
3) **Order Placement:**
   - User adds food to the cart -> Proceeds to checkout -> Order saved in Sanity.
4) **Shipment Tracking:**
   - Order status updates are fetched via a third-party API -> Displayed on the site.

## Workflow Diagram




# 4. Plan API Requirements

# API Endpoints

1) **Products API**
   - **Endpoint Name:** /products

- **Method:** GET
- **Description:** Fetch all available food items from the Sanity CMS.
- **Response Example:** [{"id": "1", "name": "Margherita Pizza", "price": 799, "stock": "Available", "image": "pizza.jpg"}, {"id": "2", "name": "Chicken Burger", "price": 299, "stock": "Out of Stock", "image": "burger.jpg"}]

### 2) Orders API

- **Endpoint Name:** /orders
- **Method:** POST
- **Description:** Create a new order in the Sanity CMS.
- **Payload Example:** {"customerId": "123", "orderItems": [{"productId": "1", "quantity": 2}, {"productId": "2", "quantity": 1}], "totalAmount": 1397, "paymentStatus": "Paid"}
- **Response Example:** {"orderId": "456", "status": "Success", "message": "Order placed successfully."}

### 3) Shipment API

- **Endpoint Name:** /shipment
- **Method:** GET
- **Description:** Track the shipment status of an order using a third-party API.
- **Response Example:** {"shipmentId": "789", "orderId": "456", "status": "In Transit", "expectedDelivery": "2025-01-18T15:00:00Z"}

## 5. Sanity CMS Schemas

- **Product Schema**

```
import { defineField, defineType } from 'sanity'
export default defineType({
  name: 'product',
  title: 'Product',
  type: 'document',
  fields: [
    defineField({
      name: 'name',
      title: 'Product Name',
      type: 'string',
      validation: Rule => Rule.required(),
    }),
```

```
defineField({
  name: 'description',
  title: 'Description',
  type: 'text',
}),
defineField({
  name: 'price',
  title: 'Price',
  type: 'number',
  validation: Rule => Rule.required(),
}),
defineField({
  name: 'category',
  title: 'Category',
  type: 'string',
  validation: Rule => Rule.required(),
}),
defineField({
  name: 'imageUrl',
  title: 'Image URL',
  type: 'url',
  validation: Rule => Rule.required(),
}),
defineField({
  name: 'available',
  title: 'Available',
  type: 'boolean',
  initialValue: true,
}),
defineField({
  name: 'ratings',
  title: 'Ratings',
  type: 'number',
  initialValue: 0,
}),
defineField({
  name: 'reviews',
  title: 'Reviews',
  type: 'array',
  of: [
    defineType({
      name: 'review',
      type: 'object',
      fields: [
```

```
          defineField({
            name: 'user',
            title: 'Username',
            type: 'string',
            validation: Rule => Rule.required(),
          }),
          defineField({
            name: 'rating',
            title: 'Rating',
            type: 'number',
            validation: Rule => Rule.required(),
          }),
          defineField({
            name: 'comment',
            title: 'Comment',
            type: 'text',
          }),],}),], }),],
})
```

- **Order Schema**

```
import { defineField, defineType } from 'sanity'
export default defineType({
  name: 'order',
  title: 'Order',
  type: 'document',
  fields: [
    defineField({
      name: 'userId',
      title: 'User ID',
      type: 'string',
      validation: Rule => Rule.required(),
    }),
    defineField({
      name: 'products',
      title: 'Products',
      type: 'array',
      of: [
        defineType({
          name: 'orderItem',
          type: 'object',
          fields: [
```

```
        defineField({
          name: 'productId',
          title: 'Product ID',
          type: 'string',
          validation: Rule => Rule.required(),
        }),
        defineField({
          name: 'quantity',
          title: 'Quantity',
          type: 'number',
          validation: Rule => Rule.required(),
        }),
        defineField({
          name: 'price',
          title: 'Price',
          type: 'number',
          validation: Rule => Rule.required(),
        }),
      ],
    }),
  ],
}),
defineField({
  name: 'totalPrice',
  title: 'Total Price',
  type: 'number',
  validation: Rule => Rule.required(),
}),
defineField({
  name: 'deliveryAddress',
  title: 'Delivery Address',
  type: 'string',
  validation: Rule => Rule.required(),
}),
defineField({
  name: 'orderStatus',
  title: 'Order Status',
  type: 'string',
  validation: Rule => Rule.required(),
}),
defineField({
  name: 'paymentStatus',
  title: 'Payment Status',
  type: 'string',
```

```
    validation: Rule => Rule.required(),
   }),
   defineField({
    name: 'paymentMethod',
    title: 'Payment Method',
    type: 'string',
   }),
   defineField({
    name: 'orderDate',
    title: 'Order Date',
    type: 'datetime',
    validation: Rule => Rule.required(),
   }),
   defineField({
    name: 'deliveryDate',
    title: 'Delivery Date',
    type: 'datetime',
   }),
  ],
})
```

***The END***

**Prepared by: Mahnoor M. Ayub**