"Marketplace Technical Foundation - [Foodtuck]"

Q-Commerce

Overview

FoodTuck is a Q-Commerce platform focused on restaurant-based food delivery. This document serves as a guide for understanding the technical foundation and implementation steps for the platform. It includes the system architecture, workflows, API requirements, and schema designs aligned with the platform's objectives.

1. Day 2 Activities: Transitioning to Technical Planning

Technical Requirements:

Frontend Requirements:

- **User-Friendly Interface:**

- Intuitive navigation for browsing restaurants and food items.

- **Responsive Design:**

- Optimized for mobile, tablet, and desktop devices.

- **Essential Pages:**

- Home, Product Listing, Product Details, Cart, Checkout, and Order Confirmation.

Backend (Sanity CMS):

- **Data Management:**

- Products, orders, customers, and promotional banners.

- **Schemas:**

- Tailored schemas for scalability and efficiency.

Third-Party APIs:

- **Payment Gateway:** Secure and seamless payment processing.

- **Shipment Tracking API:** Real-time updates on delivery status.

System Architecture

Diagram:

```
[Frontend (Next.js)]
|
[Sanity CMS] ---> [Product Data API]
|
[Third-Party APIs] ---> [Shipment Tracking API]
---> [Payment Gateway]
```

Data Flow:

1. Customers browse food items on the frontend.
2. Product details are fetched from Sanity CMS.
3. Orders are processed via the checkout system and stored in Sanity CMS.
4. Delivery updates are retrieved using the shipment tracking API.

Key Workflows

1. User Registration

- Users sign up with their email and password.
- Details are stored in Sanity CMS and a confirmation is sent.

2. Browsing Menu Items

- Customers view food categories and items.
- Data is dynamically fetched and displayed via Sanity CMS.

3. Placing an Order

- Items added to the cart are reviewed at checkout.
- Orders are processed through the payment gateway and stored in CMS.

4. Delivery Tracking

- Real-time delivery status is fetched via a shipment tracking API.

API Requirements

Endpoint	Method	Purpose	Response Example
`/products`	GET	Fetch all available food items	`{ "id": 1, "name": "Burger", "price": 150 }`
`/categories`	GET	Fetch available food categories	`{ "id": 1, "name": "Pizzas" }`
`/cart`	POST	Add an item to the cart	`{ "cartId": 456, "items": [{ "id": 1, "name": "Burger" }] }`
`/cart`	GET	Retrieve current cart details	`{ "cartId": 456, "items": [{ "id": 1, "name": "Burger" }] }`
`/checkout`	POST	Process the cart and create an order	`{ "orderId": 123, "status": "Order Placed" }`
`/orders`	POST	Create a new order	`{ "orderId": 123, "status": "Success" }`
`/express-delivery-status`	GET	Fetch delivery updates	`{ "orderId": 123, "status": "Out for Delivery" }`

Sanity CMS Schema Designs

Products Schema:

```typescript
export default {
```

name: 'product',

type: 'document',

fields: [

{ name: 'name', type: 'string', title: 'Dish Name' },

{ name: 'price', type: 'number', title: 'Price' },

{ name: 'image', type: 'image', title: 'Dish Image' },],

};

```

##### Orders Schema:

```typescript

export default {

name: 'order',

type: 'document',

fields: [

{ name: 'orderId', type: 'string', title: 'Order ID' },

{ name: 'customerName', type: 'string', title: 'Customer Name' },

{ name: 'address', type: 'string', title: 'Delivery Address' },

```
{
name: 'items',
type: 'array',
of: [{ type: 'reference', to: [{ type: 'product' }] }],
title: 'Ordered Items',
},
{ name: 'totalAmount', type: 'number', title: 'Total Amount' },
{ name: 'status', type: 'string', title: 'Order Status' },
],
};
```

Technical Roadmap

Phase 1:

- Finalize system architecture and data schemas.

Phase 2:

- Develop frontend pages and integrate Sanity CMS.

Phase 3:

- Implement checkout, payment gateway, and shipment tracking.

Phase 4:

- Conduct end-to-end testing and deploy the platform.

Conclusion

This Document serves as a comprehensive guide for understanding the technical foundation of FoodTuck. By following the outlined steps, the platform ensures seamless user experience, scalability, and effective data management. For further details, refer to the attached diagrams and schema files.