Marketplace Technical Foundation

**QUICK GROCERY**

Quarter 2

Hackathon 3

Task: Day 2

Student Umair Ahmed Siddiqui

Last Updated: 7th February, 2025

Table of Contents

[1. Introduction 3](#_Toc189839045)

[2. Technical Requirements 3](#_Toc189839046)

[Frontend Requirements 3](#_Toc189839047)

[Backend Requirements 3](#_Toc189839048)

[Third-Party APIs 3](#_Toc189839049)

[3. System Architecture 4](#_Toc189839050)

[High-Level Architecture Diagram 4](#_Toc189839051)

[Workflow Descriptions 4](#_Toc189839052)

[4. Data Schema 5](#_Toc189839053)

[Entities and Attributes: 5](#_Toc189839054)

[5. API Requirements 6](#_Toc189839055)

[Products API: 6](#_Toc189839056)

[Orders API: 6](#_Toc189839057)

[Delivery API: 6](#_Toc189839058)

[6. Sanity Schema 7](#_Toc189839059)

[Schema for Products 7](#_Toc189839060)

[Schema for Stores 7](#_Toc189839061)

[Schema for Customers 7](#_Toc189839062)

[Schema for Orders 8](#_Toc189839063)

[Schema for Order Items 8](#_Toc189839064)

[7. Key Workflows 9](#_Toc189839065)

[1. Product Browsing Workflow: 9](#_Toc189839066)

[2. Order Placement Workflow: 9](#_Toc189839067)

[3. Delivery Tracking Workflow: 9](#_Toc189839068)

# 1. Introduction

The Quick Grocery Q-Commerce store aims to provide customers with real-time inventory updates from local departmental stores and fast delivery services using a trusted riding network. This document outlines the technical foundation for the platform, including system architecture, workflows, and API requirements.

# 2. Technical Requirements

## Frontend Requirements

1. User Interface:

* A dynamic and responsive design.
* Essential pages: Home, Product List, Product Details, Cart, Checkout, Order Tracking.

1. Key Features:

* Real-time product inventory display.
* Order tracking with live rider location.
* Search and filter functionality.

1. Technology Stack:

* Framework: Next.js.
* Styling: Tailwind CSS.

## Backend Requirements

1. Data Management:

* Sanity CMS for managing product, order, and customer data.
* Integration with local departmental stores for inventory updates.

1. APIs:

* Payment gateway API for secure transactions.
* Delivery service API for rider tracking.

## Third-Party APIs

1. Store Inventory API: Provides real-time stock updates.
2. Delivery API: Tracks rider location and delivery status.
3. Payment API: Handles secure transactions.

# 3. System Architecture

## High-Level Architecture Diagram

**[Frontend (Next.js)]**

**|**

**[Sanity CMS] ---------> [Store Inventory API]**

**| |**

**[Delivery API] [Payment Gateway]**

## Workflow Descriptions

1. Product Browsing:

* Customers view products from Sanity CMS, updated via the Store Inventory API.

1. Order Placement:

* Order details are saved in Sanity CMS, and payment is processed through the Payment Gateway.

1. Delivery Tracking:

* Delivery status is updated in real-time via the Delivery API.

# 4. Data Schema

## Entities and Attributes:

1. Customer:

* customer\_id (PK)
* name
* email
* phone\_number
* address
* total\_spent

1. Order:

* order\_id (PK)
* customer\_id (FK)
* order\_status
* order\_total
* payment\_status
* created\_at
* updated\_at

1. Product:

* product\_id (PK)
* product\_name
* price
* quantity
* store\_id (FK)
* size
* weight

1. Store:

* store\_id (PK)
* store\_name
* location

1. OrderItem:

* order\_id (FK)
* product\_id (FK)
* quantity
* price\_per\_item

Note: ‘PK’ stands for Primary Key and ‘FK’ stands for Foreign Key.

# 5. API Requirements

## Products API:

* Endpoint: `/products`
* Method: GET
* Description: Fetches available grocery items.
* Response Example:

{

"id": 101,

"name": "Milk (1L)",

"price": 2.5,

"stock": 15,

"storeId": 3,

"lastUpdated": "2025-01-20T12:00:00Z"

}

## Orders API:

* Endpoint: `/orders`
* Method: POST
* Description: Saves a new order.
* Payload Example:

{

"customerId": 12,

"products": [

{"productId": 101, "quantity": 2},

{"productId": 102, "quantity": 1}

],

"totalAmount": 15.5

}

## Delivery API:

* Endpoint: `/rider`
* Method: GET
* Description: Fetches real-time rider location.
* Response Example:

{

"riderId": 5,

"status": "En Route",

"currentLocation": "Main Street, Block A",

"ETA": "10 minutes"

}

# 6. Sanity Schema

## Schema for Products

export default {

name: 'product',

type: 'document',

title: 'Product',

fields: [

{ name: 'productName', type: 'string', title: 'Product Name' },

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

{ name: 'quantity', type: 'number', title: 'Stock Quantity' },

{ name: 'storeId', type: 'reference', to: [{ type: 'store' }], title: 'Store' },

{ name: 'size', type: 'string', title: 'Size (optional)' },

{ name: 'weight', type: 'string', title: 'Weight (optional)' },

{ name: 'lastUpdated', type: 'datetime', title: 'Last Updated' }

]

};

## Schema for Stores

export default {

name: 'store',

type: 'document',

title: 'Store',

fields: [

{ name: 'storeName', type: 'string', title: 'Store Name' },

{ name: 'location', type: 'string', title: 'Location' }

]

};

## Schema for Customers

export default {

name: 'customer',

type: 'document',

title: 'Customer',

fields: [

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

{ name: 'email', type: 'string', title: 'Email' },

{ name: 'phoneNumber', type: 'string', title: 'Phone Number' },

{ name: 'address', type: 'text', title: 'Address' },

{ name: 'totalSpent', type: 'number', title: 'Total Spent' }

]

};

## Schema for Orders

export default {

name: 'order',

type: 'document',

title: 'Order',

fields: [

{ name: 'customerId', type: 'reference', to: [{ type: 'customer' }], title: 'Customer' },

{ name: 'orderStatus', type: 'string', title: 'Order Status', options: { list: ['Pending', 'Shipped', 'Delivered', 'Cancelled'] } },

{ name: 'orderTotal', type: 'number', title: 'Total Amount' },

{ name: 'paymentStatus', type: 'string', title: 'Payment Status', options: { list: ['Paid', 'Pending', 'Failed'] } },

{ name: 'createdAt', type: 'datetime', title: 'Order Created At' },

{ name: 'updatedAt', type: 'datetime', title: 'Last Updated' }

]

};

## Schema for Order Items

export default {

name: 'orderItem',

type: 'document',

title: 'Order Item',

fields: [

{ name: 'orderId', type: 'reference', to: [{ type: 'order' }], title: 'Order' },

{ name: 'productId', type: 'reference', to: [{ type: 'product' }], title: 'Product' },

{ name: 'quantity', type: 'number', title: 'Quantity' },

{ name: 'pricePerItem', type: 'number', title: 'Price Per Item' }

]

};

# 7. Key Workflows

## Product Browsing Workflow:

## Order Placement Workflow:

## Delivery Tracking Workflow: