Auto Grocery Microservices

API Technical Specification

Ordering Service (v1.0)

Backend Architecture Team

February 10, 2026

Chapter 1

Architecture Overview

1.1 Service Responsibility

The **Ordering Service** is the primary gateway for all external interactions. It is designed as a stateless orchestrator that manages the lifecycle of a grocery order.

- **Port:** `5050`
- **Database:** PostgreSQL (Stores User Accounts & Order History)
- **Downstream Dependencies:**
  - **Inventory Service (gRPC):** Handles stock reservations and atomic decrements.
  - **Pricing Service (gRPC):** Handles dynamic pricing calculations based on batch costs.

1.2 Authentication Mechanism

The system implements a stateless **Dual-Token JWT Architecture**.

Access Token:
- **Format:** JWT (RS256 Signature)
- **Expiry:** 15 Minutes
- **Header:** `Authorization: Bearer <token>`
- **Claim `token_type`:** Must be `"ACCESS"`.

Refresh Token:
- **Format:** JWT (RS256 Signature)
- **Expiry:** 7 Days
- **Usage:** Exchanged at `/api/client/refresh` only.
- **Claim `token_type`:** Must be `"REFRESH"`.

Chapter 2

Client API Reference

Base URL: ***http://localhost:5050***

POST /api/client/register

Access Level: Public

Description: Onboards a new user or smart fridge device.

Request Schema

Field	Type	Req	Description
device_id	String	Yes	Unique hardware ID or username. Max 50 chars.
password	String	Yes	Minimum 8 characters.
email	String	No	Contact email.
phone	String	No	Contact phone number.

Internal Execution Flow

1. **Validation:** Checks if `device_id` is not empty.
2. **Sanitization:** Trims whitespace from inputs.
3. **Hashing:** Uses `bcrypt` to hash the password (Cost: 10).
4. **Persistence:** SQL `INSERT` into `smart_devices` table.
5. **Error Handling:** If `device_id` exists, returns HTTP 409 Conflict.

Example

```
// REQUEST
{
  "device_id": "user_john_doe",
  "password": "securePass123!",
  "email": "john@example.com"
}

// RESPONSE (201 Created)
{
  "status": "success",
  "message": "User registered successfully"
}
```

POST `/api/client/login`

Access Level: Public

Description: Authenticates a user and issues a generic session.

Request Schema

Field	Type	Req	Description
`device_id`	String	Yes	Registered ID.
`password`	String	Yes	Plaintext password.

Internal Execution Flow

1. **Lookup:** SELECT query on `smart_devices` by `device_id`.
2. **Verification:** `bcrypt.CompareHashAndPassword` checks validity.
3. **Token Generation:**
   - Signs `AccessToken` with `type="ACCESS"` (Exp: 15m).
   - Signs `RefreshToken` with `type="REFRESH"` (Exp: 7d).
4. **Response:** Returns both tokens.

Example

```
// RESPONSE (200 OK)
{
  "access_token": "eyJhbGciOiJIUzI1NiIsIn...",
  "refresh_token": "eyJhbGciOiJIUzI1NiIsIn..."
}
```

POST `/api/client/order/preview`

Access Level: Authenticated (Bearer)

Description: The "Add to Cart" action. It reserves inventory to prevent overselling.

Request Schema

Field	Type	Req	Description
`items`	Array	Yes	List of items to order.
`items[].sku`	String	Yes	Stock Keeping Unit (e.g., "APPLE").
`items[].qty`	Int	Yes	Must be > 0.

Internal Execution Flow (The "Reserve" Transaction)

1. **Middleware:** Extracts `user_id` from JWT. Fails if token expired.
2. **gRPC Call (Inventory):** Calls `ReserveStock(items)`.
   - Inventory Service checks stock levels.
   - If stock exists, it moves items to "Reserved" state.
   - If stock is low, returns `gRPC Status: FAILED_PRECONDITION`.
3. **Order Creation:** Logic creates a new row in `grocery_orders`.
4. **Status Setting:** Order status set to `"RESERVED"`.
5. **ID Generation:** Generates UUID (e.g., `980fa...`).

Example

```
// REQUEST
{
  "items": [
    { "sku": "APPLE", "quantity": 5 },
    { "sku": "BANANA", "quantity": 10 }
  ]
}

// RESPONSE (200 OK)
{
  "order_id": "980fa849-7093-4c1c-aa32-f6c648f2c6ba",
  "status": "reserved",
  "items": { "APPLE": 5, "BANANA": 10 }
}
```

POST `/api/client/order/confirm`

Access Level: Authenticated (Bearer)

Description: Finalizes the purchase. This is the "Checkout" button. It triggers pricing calculation.

Request Schema

Field	Type	Req	Description
`order_id`	UUID	Yes	The ID returned from the Preview step.

Internal Execution Flow

1. **Retrieval:** Fetches Order #ID from Postgres.
2. **State Guard:** Checks if Order Status is `"RESERVED"`.
    - If `"COMPLETED"`, returns Error 400 (Already paid).
    - If `"CANCELLED"`, returns Error 400.
3. **gRPC Call (Pricing):** Calls `GetPrice(items)`.
    - Pricing Service calculates cost based on current batch + margin.
4. **Completion:**
    - Updates Postgres: `status = "COMPLETED"`.
    - Updates Postgres: `total_price = <value>`.

Example

```
// REQUEST
{ "order_id": "980fa849-7093-4c1c-aa32-f6c648f2c6ba" }

// RESPONSE (200 OK)
{
  "order_id": "980fa849-7093-4c1c-aa32-f6c648f2c6ba",
  "status": "completed",
  "total_price": 3.00,
  "timestamp": "2026-02-09T02:02:38Z"
}
```

POST `/api/client/order/cancel`

Access Level: Authenticated (Bearer)

Description: Cancels a reservation and releases stock back to the general pool.

Internal Execution Flow

1. **Retrieval:** Fetches Order #ID.
2. **gRPC Call (Inventory):** Calls `ReleaseStock(items)`.
   - Inventory Service increments available stock.
3. **Update:** Sets status to `"CANCELLED"`.

Chapter 3

Truck Logistics API

POST `/api/truck/restock`

Access Level: Authenticated (Truck Role)

Description: Used by smart trucks or robots to offload items into the warehouse.

Request Schema

Field	Type	Req	Description
`truck_id`	String	Yes	License plate or Robot ID.
`supplier_id`	String	Yes	Origin of goods.
`items`	Array	Yes	List of goods.
`items[].unit_cost`	Float	Yes	**Crucial**: The cost price for this batch.

Internal Execution Flow

1. **Audit Log:** Inserts record into `restock_orders` table (for tracking).
2. **gRPC Call (Inventory):** Calls `RestockItems(items)`.
   - Inventory adds quantity to `stock` table.
   - Inventory records the new `unit_cost` for pricing.
3. **Trigger:** Inventory Service may asynchronously notify Pricing Service (via gRPC Trigger).

Example

```
// REQUEST
{
  "truck_id": "T-1000",
  "supplier_id": "OrganicFarms",
  "items": [
    {
      "sku": "APPLE",
      "quantity": 100,
      "unit_cost": 0.50,
      "mfd_date": "2026-02-01",
      "expiry_date": "2026-03-01"
    }
  ]
}

// RESPONSE (201 Created)
{
  "status": "success",
  "message": "Inventory updated",
  "transaction_id": "tx_unique_123"
}
```