Restaurant API Technical Documentation

Group 10

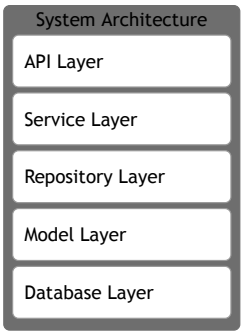
August 6, 2025

API Reference

This API adheres to RESTful architecture. It is a work in progress; not all features are fully implemented.

System Architecture

The Restaurant API is built using a **layered architecture** pattern with clear separation of concerns:



Development Environment Setup

Required Tools

- Python 3.8+
- MySQL
- pip package manager

Setup Instructions

```
# 1. Install dependencies
pip install -r requirements.txt

# 2. Configure database
Modify api/dependencies/config.py to match your database

# 4. Start server
uvicorn api.main:app --reload

# 5. Add test data
python scripts/add_test_customers.py
python scripts/add_test_food.py
python scripts/add_test_promotions.py
```

Code Examples & Implementation Details

The API uses cookie-based session management for cart persistence, this is a temporary solution for development:

api/routers/customer_actions.py

```python
# How cookies are READ:
order_id: Optional[int] = Cookie(None)

# How cookies are SET:
response.set_cookie(key="order_id", value=str(result["order_id"]), max_age=3600)
```

HTTP Status Codes

The API uses standard HTTP status codes to indicate the success or failure of requests. Most error responses include error messages to help with debugging.

HTTP Status Codes

```
200 OK
201 Created
400 Bad Request
404 Not Found
409 Conflict
422 Unprocessable Entity
500 Internal Server Error
```

API Endpoint Summary

Customer Actions

Customer-facing endpoints for browsing menu, placing orders, and tracking deliveries. These endpoints handle the complete customer journey from menu browsing to order completion.

ENDPOINTS

```
GET    /customer/menu
GET    /customer/menu/{item_name}
POST   /customer/add-to-cart
DELETE /customer/remove-from-cart
POST   /customer/choose-order-type
POST   /customer/add-customer-information
POST   /customer/add-payment
POST   /customer/add-promo-code-to-order
POST   /customer/checkout
GET    /customer/track-order/{tracking_number}
POST   /customer/review-dish
GET    /customer/view-reviews
```

Staff Actions

Staff management endpoints for order processing, inventory management, and menu item creation. These endpoints allow restaurant staff to manage daily operations.

ENDPOINTS

```
GET    /staff_actions/view-ingredients
GET    /staff_actions/promotions/code/{promo_code}
PUT    /staff_actions/promotions/code/{promo_code}
DELETE /staff_actions/promotions/code/{promo_code}
GET    /staff_actions/revenue/{target_date}
GET    /staff_actions/reviews/{rating}
GET    /staff_actions/orders/status/{status}
PUT    /staff_actions/orders/{order_id}/status
GET    /staff_actions/orders/date-range
POST   /staff_actions/add-menu-item
PUT    /staff_actions/update-stock
```

Analytics

Business intelligence endpoints for analyzing restaurant performance, dish popularity, and customer feedback trends over different time periods.

ENDPOINTS

```
GET    /analytics/dish-analytics-average-rating
GET    /analytics/dish-analytics-popularity
GET    /analytics/view-reviews
```

Administrator Actions

Administrative endpoints for database management and test data creation. These endpoints are used for system maintenance and development purposes.

ENDPOINTS

```
POST   /admin/add-test-data/{data_type}
DELETE /admin/purge-db
```

Core Objects

The Customer Object

Attributes

id	int
customer_name	string
customer_email	string
customer_phone	int
customer_address	string

The Customer Object


```
{
  "id": 1,
  "customer_name": "Jason Bourne",
  "customer_email": "jason.bourne@email.com",
  "customer_phone": 1234567,
  "customer_address": "Blue Ridge Parkway, Asheville, NC 28801"
}
```


The Order Object

Attributes

id	int
customer_id	int
order_date	datetime
description	string
status	enum

pending, confirmed, in_progress, awaiting_pickup, out_for_delivery, cancelled, completed

order_type	enum

dine_in, takeout, delivery

promo_id	int
paid	boolean
tracking_number	string

The Order Object


```
{
  "id": 123,
  "customer_id": 1,
  "order_date": "2025-08-08T14:30:00",
  "description": "Example order 1",
  "status": "completed",
  "order_type": "dine_in",
  "promo_id": null,
  "paid": true,
  "tracking_number": null
}
```


The MenuItem Object

Attributes

id	int
name	string
description	string
price	float

The MenuItem Object


```
{
  "id": 1,
  "name": "ExtravaganZZa Pizza",
  "description": "Pepperoni, ham, Italian sausage, beef, onions,
  "price": 16.99,
  "calories": 320,
  "food_category": "regular"
}
```

`calories` int

`food_category` enum

vegetarian, vegan, gluten_free, regular

The Review Object

Attributes

`id` int

`menu_item_id` int

`customer_name` string

`rating` int

Rating from 1 to 5

`review_text` string

`created_at` datetime

The Review Object


```
{
  "id": 1,
  "menu_item_id": 1,
  "customer_name": "Jason Bourne",
  "rating": 5,
  "review_text": "Jesus Christ, it's...",
  "created_at": "2025-08-08T12:00:00"
}
```


The Promotion Object

Attributes

`id` int

`code` string

`description` string

`discount_percent` int

Percentage from 1 to 100

`expiration_date` datetime

`created_at` datetime

The Promotion Object


```
{
  "id": 1,
  "code": "SAVE10",
  "description": "Save 10% on your order",
  "discount_percent": 10,
  "expiration_date": "2025-09-07T23:59:59",
  "created_at": "2025-08-08T12:00:00"
}
```


The Payment Object

Attributes

`id` int

`order_id` int

`payment_date` datetime

`status` enum

pending, completed, failed, refunded

`payment_type` enum

cash, credit_card, debit_card

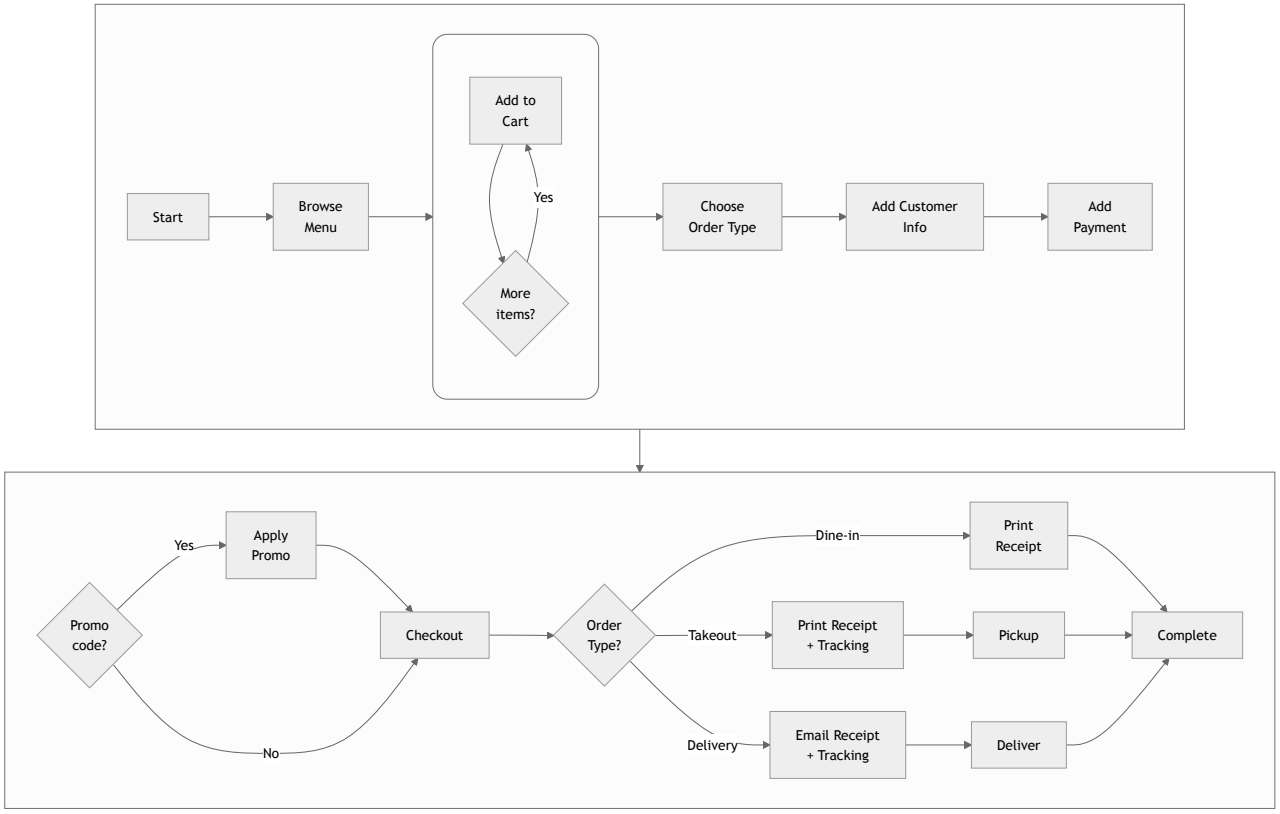
`card_number` string

The Payment Object


```
{
  "id": 1,
  "order_id": 123,
  "payment_date": "2025-08-08T14:35:00",
  "status": "completed",
  "payment_type": "cash",
  "card_number": null
}
```

The Customer Ordering Process

The customer ordering process begins when a user browses the menu and adds desired items to their cart, repeating until all selections are made. After choosing the order type (dine-in, takeout, or delivery), the customer provides contact and payment details. If a promo code is available, it can be applied before checkout. The system then finalizes the order, processes payment, and generates a receipt or tracking number based on the selected order type.



Customer Actions Endpoints

Get Available Menu

Retrieves all available menu items based on ingredient availability. Items are only shown to the customer if all required ingredients are in stock.

Parameters

`filter_category` string optional

Filter by food category: vegetarian, vegan, gluten_free, regular

Returns

Array of MenuItem objects with availability status

```
GET /customer/menu
```


```
[
  {
    "id": 1,
    "name": "ExtravaganZZa Pizza",
    "description": "Pepperoni, ham, Italian sausage, beef, onions
    "price": 16.99,
    "calories": 320,
    "food_category": "regular"
  },
  {
    "id": 7,
    "name": "Chicken Caesar Salad",
    "description": "Caesar salad topped with grilled chicken bred
    "price": 9.99,
    "calories": 280,
    "food_category": "regular"
  }
]
```

Get Menu Item by Name

Search for a specific menu item by name. Returns the item only if it's available (has sufficient ingredients).

Parameters

`item_name` string required

Name or partial name of the menu item to search for

Returns

A MenuItem object if found and available

```
GET /customer/menu/{item_name}
```


```
{
  "id": 1,
  "name": "ExtravaganZZa Pizza",
  "description": "Pepperoni, ham, Italian sausage, beef, onions,
  "price": 16.99,
  "calories": 320,
  "food_category": "regular"
}
```


Add Item to Cart

Adds a menu item to the customer's cart. Creates a new order if none exists, or adds to existing order. Sets order_id cookie for session management.

Parameters

`menu_item_id` int required

`quantity` int required

`customer_id` int optional

Returns

Success message with order_id

```
Request Body
```


```
{
  "menu_item_id": 1,
  "quantity": 2
}
```


```
POST /customer/add-to-cart
```


```
{
  "message": "ExtravaganZZa Pizza added to cart",
  "order_id": 123
}
```


Remove Item from Cart

Removes a menu item completely from the customer's cart. Requires active order session.

Parameters

`menu_item_id` int required

ID of the menu item to remove from cart

Returns

Success message confirming item removal

```
DELETE /customer/remove-from-cart
```


```
{
  "message": "Margherita Pizza removed from cart"
}
```

Choose Order Type

Sets the order type for the current order. Affects pricing, tracking, and fulfillment process.

Parameters

`type` enum required

Order type: dine_in, takeout, delivery

Returns

Updated Order object with new order type

POST /customer/choose-order-type

```
{
  "type": "delivery"
}
```

Successful Response

```
{
  "id": 123,
  "customer_id": null,
  "order_date": null,
  "description": "Customer cart",
  "status": "pending",
  "order_type": "delivery",
  "promo_id": null,
  "paid": false,
  "tracking_number": null
}
```

Add Customer Information

Adds customer information to the current order. Creates new customer or updates existing one based on email/phone.

Parameters

`customer_name` string required

`customer_email` string optional

`customer_phone` int optional

`customer_address` string optional

Returns

Updated Order object with customer information

POST /customer/add-customer-information

```
{
  "customer_name": "Sarah Johnson",
  "customer_email": "sarah.johnson@email.com",
  "customer_phone": 5551234567,
  "customer_address": "456 Oak Avenue, Springfield, IL 62701"
}
```

Successful Response

```
{
  "id": 123,
  "customer_id": 1,
  "order_date": null,
  "description": "Customer cart",
  "status": "pending",
  "order_type": "delivery",
  "promo_id": null,
  "paid": false,
  "tracking_number": null
}
```

Add Payment Method

Adds payment method to the current order. Card number is required for card payments but ignored for cash.

Parameters

`payment_type` enum required

Payment method: cash, credit_card, debit_card

`card_number` string optional

Required for card payments, ignored for cash

Returns

Payment object with pending status

POST /customer/add-payment

```
{
  "payment_type": "credit_card",
  "card_number": "4532-1234-5678-9012"
}
```

Successful Response

```
{
  "id": 89,
  "order_id": 123,
  "payment_date": "2025-08-08T14:35:00",
  "status": "pending",
  "payment_type": "credit_card",
  "card_number": "****-****-****-9012"
}
```

Add Promo Code

Applies a promotional code to the current order. Validates code existence and expiration before applying.

Parameters

`promo_code` string required

Valid promotional code to apply to the order

Returns

Updated Order object with applied promotion

POST /customer/add-promo-code-to-order

```
{
  "promo_code": "SAVE10"
}
```

Successful Response

```
{
  "id": 123,
  "customer_id": 1,
  "order_date": null,
  "description": "Customer cart",
  "status": "pending",
  "order_type": "delivery",
  "promo_id": 1,
  "paid": false,
  "tracking_number": null
}
```

Checkout Order

Processes the order checkout. Calculates totals, processes payment, updates inventory, and generates tracking number for takeout/delivery orders.

Parameters

None. Uses order_id from session cookie.

Returns

Checkout confirmation with total and tracking number (if applicable)

POST /customer/checkout Response

```
{
  "message": "Order successfully created",
  "total": 42.67,
  "tracking_number": "TRK8A9B2"
}
```

Successful Response

```
{
  "message": "Order successfully created",
  "total": 42.67
}
```

Track Order

Retrieves order status and details using the tracking number provided at checkout.

Parameters

`tracking_number` string required

8-character tracking number from checkout

Returns

Order tracking information with current status

GET /customer/track-order/{tracking_number}


```
{
  "tracking_number": "TRK8A9B2",
  "order_id": 123,
  "status": "in_progress",
  "order_type": "delivery",
  "order_date": "2025-08-08T14:30:00",
  "paid": true
}
```


Add Review

Allows customers to add a review and rating for a menu item they've tried.

Parameters

`menu_item_id` int required

`customer_name` string required

`rating` int required

Rating from 1 to 5 stars

`review_text` string optional

Returns

Created Review object with timestamp

POST /customer/review-dish


```
{
  "menu_item_id": 1,
  "customer_name": "Jason Bourne",
  "rating": 5,
  "review_text": "Oh no, it's..."
}
```


Successful Response


```
{
  "id": 1,
  "menu_item_id": 1,
  "customer_name": "Jason Bourne",
  "rating": 5,
  "review_text": "Here he comes, it's...",
  "created_at": "2025-08-08T12:00:00"
}
```


View Reviews

Retrieves all menu items that have customer reviews along with their review details.

Parameters

No parameters required.

Returns

Array of menu items with their associated reviews

GET /customer/view-reviews


```
[
  {
    "menu_item_name": "ExtravaganZZa Pizza",
    "reviews": [
      {
        "customer_name": "Jason Bourne",
        "rating": 5,
        "review_text": "Uh oh, it's...",
        "created_at": "2025-08-08T12:00:00"
      },
      {
        "customer_name": "Peter Peterson",
        "rating": 3,
        "review_text": "It was okay.",
        "created_at": "2025-08-08T12:00:00"
      }
    ]
  }
]
```

Staff Actions Endpoints

View Ingredients

Retrieves the required ingredients and quantities needed to prepare a specific menu item. Used for inventory planning and stock verification.

Parameters

`menu_item_id` int required

ID of the menu item to check ingredients for

`quantity` int required

Number of servings to calculate ingredients for

Returns

An array of ingredient objects with required quantities and current stock levels if a valid menu item ID is provided.

Raises a 404 error if the menu item is not found.

```
GET /staff_actions/view-ingredients?menu_item_id=1&quantity=2
```


```
[
  {
    "resource_name": "Pizza Dough",
    "required_quantity": 2,
    "unit": "pieces",
    "available_stock": 10
  },
  {
    "resource_name": "Pizza Sauce",
    "required_quantity": 6,
    "unit": "oz",
    "available_stock": 10
  },
  {
    "resource_name": "Mozzarella Cheese",
    "required_quantity": 8,
    "unit": "oz",
    "available_stock": 10
  }
]
```


Get Promotion by Code

Retrieves promotion details by promotional code. Used to verify and apply discounts to customer orders.

Parameters

`promo_code` string required

Promotional code to look up

Returns

A promotion object with discount details if a valid promotional code is provided.

Raises a 404 error if the promotion code is not found or has expired.

```
GET /staff_actions/promotions/code/{promo_code}
```


```
{
  "id": 1,
  "code": "SAVE10",
  "description": "Save 10% on your order",
  "discount_percent": 10,
  "expiration_date": "2025-09-07T23:59:59",
  "created_at": "2025-08-08T12:00:00"
}
```


Update Promotion by Code

Updates an existing promotion's discount percentage and expiration date. Used for managing promotional campaigns.

Parameters

`promo_code` string required

Promotional code to update

`discount_percent` int required

New discount percentage (1-100)

`expiration_date` datetime optional

New expiration date for the promotion

Returns

The updated promotion object if a valid promotional code is provided and the update is successful.

Raises a 404 error if the promotion code is not found or a 422 error if the discount percentage is invalid.

```
PUT /staff_actions/promotions/code/{promo_code}
```


```
{
  "discount_percent": 15,
  "expiration_date": "2025-09-15T23:59:59"
}
```


```
Successful Response
```


```
{
  "id": 1,
  "code": "SAVE10",
  "description": "Save 10% on your order",
  "discount_percent": 15,
  "expiration_date": "2025-09-15T23:59:59",
  "created_at": "2025-08-08T12:00:00"
}
```

Delete Promotion by Code

Removes a promotion from the system. Used to deactivate expired or cancelled promotional campaigns.

Parameters

`promo_code` string required

Promotional code to delete

Returns

A success message confirming the promotion deletion if a valid promotional code is provided.

Raises a 404 error if the promotion code is not found.

```
DELETE /staff_actions/promotions/code/{promo_code}
```


```
{
  "message": "Promotion SAVE10 deleted successfully"
}
```


Get Daily Revenue

Retrieves total revenue for a specific date. Used for daily sales reporting and financial tracking.

Parameters

`target_date` date required

Date to calculate revenue for (YYYY-MM-DD format)

Returns

A revenue summary object with total amount, order count, and average order value if a valid date is provided.

Raises a 422 error if the date format is invalid.

```
GET /staff_actions/revenue/{target_date}
```


```
{
  "date": "2025-08-08",
  "total_revenue": 1247.85,
  "total_orders": 23,
  "average_order_value": 54.25
}
```


Get Reviews by Rating

Retrieves menu items and their reviews filtered by a specific rating. Used for quality control and menu optimization.

Parameters

`rating` int required

Rating value between 1 and 5

Returns

An array of menu items with their associated reviews matching the specified rating if a valid rating (1-5) is provided.

Raises a 422 error if the rating is outside the valid range.

```
GET /staff_actions/reviews/{rating}
```


```
[
  {
    "menu_item_name": "ExtravaganZZa Pizza",
    "menu_item_id": 1,
    "reviews": [
      {
        "id": 1,
        "customer_name": "Jason Bourne",
        "rating": 5,
        "review_text": "What do we do?, it's...",
        "created_at": "2025-08-08T12:00:00"
      }
    ]
  }
]
```

Get Orders by Status

Retrieves all orders filtered by their current status. Used for order management and kitchen workflow organization.

Parameters

`status` enum required

Order status: pending, confirmed, in_progress, awaiting_pickup, out_for_delivery, cancelled, completed

Returns

An array of Order objects with the specified status if a valid status is provided.

Raises a 422 error if the status is not a valid order status.

```
GET /staff_actions/orders/status/{status}
```


```
[
  {
    "id": 123,
    "customer_id": 1,
    "order_date": "2025-08-08T14:30:00",
    "description": "Example order 1",
    "status": "in_progress",
    "order_type": "dine_in",
    "promo_id": null,
    "paid": true,
    "tracking_number": null
  }
]
```


Update Order Status

Updates the status of a specific order. Used to track order progress through the fulfillment pipeline.

Parameters

`order_id` int required

ID of the order to update

`status` enum required

New status: pending, confirmed, in_progress, awaiting_pickup, out_for_delivery, cancelled, completed

Returns

The updated Order object with new status if a valid order ID and status are provided.

Raises a 404 error if the order is not found or a 422 error if the status is invalid.

```
PUT /staff_actions/orders/{order_id}/status?status=completed
```


```
{
  "id": 123,
  "customer_id": 1,
  "order_date": "2025-08-08T14:30:00",
  "description": "Example order 1",
  "status": "completed",
  "order_type": "dine_in",
  "promo_id": null,
  "paid": true,
  "tracking_number": null
}
```


Get Orders by Date Range

Retrieves orders within a specified date range. Used for historical reporting and trend analysis.

Parameters

`start_date` date required

Start date of the range (YYYY-MM-DD format)

`end_date` date required

End date of the range (YYYY-MM-DD format)

Returns

An array of Order objects within the specified date range if valid dates are provided.

Raises a 422 error if the date format is invalid or the start date is after the end date.

```
GET /staff_actions/orders/date-range?
start_date=2025-08-01&end_date=2025-08-08
```


```
[
  {
    "id": 123,
    "customer_id": 1,
    "order_date": "2025-08-08T14:30:00",
    "description": "Example order 1",
    "status": "completed",
    "order_type": "dine_in",
    "promo_id": null,
    "paid": true,
    "tracking_number": null
  }
]
```

Add Menu Item

Creates a new menu item with its required ingredients and quantities. Used for menu expansion and seasonal offerings.

Parameters

`name` string required

`price` float required

`food_category` enum required

Category: vegetarian, vegan, gluten_free, regular

`calories` int required

`description` string optional

`resources` array required

Array of resource requirements with resource_name and quantity

Returns

A created MenuItem object with assigned ID if successful.

Raises a 422 error if required parameters are missing or invalid, or a 409 error if a menu item with the same name already exists.

```
POST /staff_actions/add-menu-item
```


```
{
  "name": "Veggie Supreme Pizza",
  "price": 15.99,
  "food_category": "vegetarian",
  "calories": 290,
  "description": "Pizza with fresh vegetables and cheese",
  "resources": [
    {
      "resource_name": "Pizza Dough",
      "quantity": 1
    },
    {
      "resource_name": "Pizza Sauce",
      "quantity": 3
    },
    {
      "resource_name": "Mozzarella Cheese",
      "quantity": 4
    }
  ]
}
```


```
Successful Response
```


```
{
  "id": 10,
  "name": "Veggie Supreme Pizza",
  "description": "Pizza with fresh vegetables and cheese",
  "price": 15.99,
  "calories": 290,
  "food_category": "vegetarian"
}
```


Update Stock

Updates inventory stock levels for a specific ingredient. Use positive numbers to add stock, negative numbers to remove stock.

Parameters

`resource_name` string required

Name of the ingredient to update

`amount_change` int required

Amount to add (positive) or remove (negative) from stock

Returns

Updated resource information with new stock level if successful.

Raises a 404 error if the resource is not found or a 422 error if the amount change would result in negative stock.

```
PUT /staff_actions/update-stock?resource_name=Mozzarella Cheese&amount_change=5
```


```
{
  "resource_name": "Mozzarella Cheese",
  "previous_stock": 10,
  "amount_change": 5,
  "new_stock": 15,
  "unit": "units"
}
```

Analytics Endpoints

Get Dish Analytics - Average Rating

Retrieves average ratings for all menu items within a specified time range. Essential for identifying top-performing dishes and menu optimization decisions.

Parameters

`time_range` enum optional

Time period for analysis: week, month, year, all_time (default: week)

Returns

Array of dishes with their average ratings for business intelligence analysis

```
GET /analytics/dish-analytics-average-rating?time_range=month
```


```
[
  {
    "dish_name": "ExtravaganZZa Pizza",
    "average_rating": 4.0
  },
  {
    "dish_name": "MeatZZa Pizza",
    "average_rating": 4.0
  }
]
```


Get Dish Analytics - Popularity

Analyzes dish popularity based on order frequency within a time range. Critical for inventory planning, menu optimization, and identifying trending items.

Parameters

`time_range` enum optional

Time period for analysis: week, month, year, all_time (default: week)

`sort_by` enum optional

Sort order: low (ascending), high (descending) (default: low)

Returns

An array of dishes with order counts sorted by popularity.

```
GET /analytics/dish-analytics-popularity?
time_range=week&sort_by=high
```


```
[
  {
    "dish_name": "ExtravaganZZa Pizza",
    "order_count": 45
  },
  {
    "dish_name": "MeatZZa Pizza",
    "order_count": 32
  },
  {
    "dish_name": "Hawaiian Pizza",
    "order_count": 28
  }
]
```


View Reviews Analytics

Retrieves all customer reviews with menu item details, sorted chronologically. Provides comprehensive feedback analysis for quality improvement and customer satisfaction tracking.

Parameters

`sort_by` enum required

Sort order: newest (most recent first), oldest (oldest first)

Returns

An array of all reviews with menu item names.

```
GET /analytics/view-reviews?sort_by=newest
```


```
[
  {
    "menu_item_name": "ExtravaganZZa Pizza",
    "rating": 5,
    "text": "RUN, it's...",
    "date": "2025-08-08T12:00:00"
  },
  {
    "menu_item_name": "MeatZZa Pizza",
    "rating": 4,
    "text": "Good, but I wanted a bigger discount",
    "date": "2025-08-08T12:00:00"
  }
]
```

Administrator Actions Endpoints

Add Test Data

Populates the database with test data for development and testing purposes.

Parameters

`data_type` enum required

Type of test data to add: customers, food, promotions, reviews, orders

Returns

Success message confirming test data creation for system development

```
POST /admin/add-test-data/{data_type}
```


```
{
  "message": "Test customers added successfully"
}
```


Purge Database

Permanently deletes all data from the database. Critical system maintenance endpoint for development environment resets and testing cleanup. **WARNING: This operation is irreversible.**

Parameters

No parameters required

Returns

Success message confirming complete database purge for system maintenance

```
DELETE /admin/purge-db
```


```
{
  "message": "Database purged successfully."
}
```