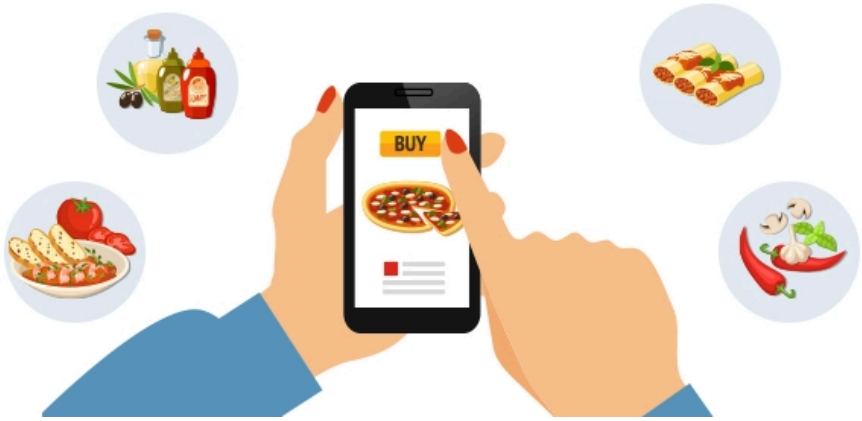


Online Restaurant Ordering System API - User Manual

12.08.2024

—

Created By

- Mason Scarbro
- Vincent Medina
- Jordan Koontz
- Elijah Collins

Overview

The Online Restaurant Ordering System API is designed to simplify the process of managing restaurant operations. This API serves as a bridge between customers, restaurant staff, and the underlying database, allowing for efficient handling of essential tasks. The API enables seamless interaction with customer data, menu items, orders, reviews, and pantry inventory, making it easier to manage day-to-day activities in a restaurant environment.

Key Features Include:

Customer Management: Easily create, update, and delete customer information.

Menu Management: Add and manage various menus and menu items dynamically.

Order Processing: Place, update, and manage customer orders in real time.

Review System: Collect and manage customer feedback to enhance service quality.

Inventory Management: Maintain pantry stock and payment information efficiently.

This API makes it easy for users to handle restaurant operations efficiently.

Setup Instructions

Prerequisites

- **Python 3.9+** installed on your machine.
- **FastAPI** framework and necessary libraries.
- **PostgreSQL** or a similar SQL database.

Installation Steps

1. **Clone the Repository:**

   git clone https://github.com/MasonScarbro/GroupProject-ITSC-3155.git

2. **Install Dependencies:**

   Requirements: fastapi, uvicorn, sqlalchemy, pymysql, pytest, pytest-mock, httpx, and cryptography

   Necessary packages:

pip install fastapi

pip install "uvicorn[standard]"

pip install sqlalchemy

pip install pymysql

pip install pytest

pip install pytest-mock

pip install httpx

pip install cryptography

3. **Set Up the Database:**

   Configure your database settings in config.py or .env file.

4. **Run the API Server:**

   uvicorn main:app --reload

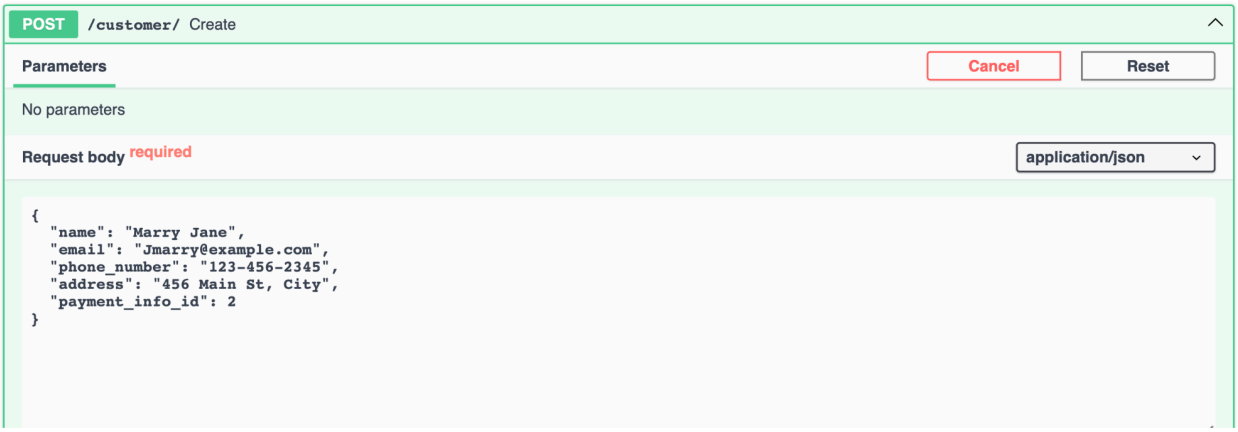
5. **Access the API Documentation:**

   Navigate to http://127.0.0.1:8000/docs for the interactive Swagger UI.

Usage Examples

Creating a Customer:

- **Endpoint: POST /customers/**
- **Request Example:**



- **Response Example:**

Fetching All Menu Items

- **Endpoint: GET /menu-items/**
- **Response Example:**


```
[
 {
  "id": 1,
  "dish": "Cheeseburger",
  "calories": 550,
  "price": 7.99,
  "menu_id": 1
 },
 {
  "id": 2,
  "dish": "Veggie Pizza",
  "calories": 450,
  "price": 9.99,
  "menu_id": 2
 }
]
```

Deleting an Order

- **Endpoint: DELETE /orders/{order_id}**
- **Request:**
  - DELETE /orders/1
- **Response:**

  {“detail": "Order deleted successfully”}

Error Handling:

404 Not Found: When a resource does not exist.

400 Bad Request: Invalid input or database error.