Restaurant API User Manual

Snack Overflow

Tevin Parathattal, Nicholas Smit, Preston Murray, Jason Springer-Trammell, Walker Thierbach

August 7, 2025

What is the Snack Overflow API?

The Snack Overflow Restaurant API is a web-based system that helps restaurants manage their entire operation online. Whether you're a restaurant owner looking to streamline operations or a developer integrating ordering capabilities into a website or mobile app, this API has everything you need.

What Can You Do With This API?

Our API helps you manage every aspect of your restaurant business:

• Take Orders: Accept orders from customers dining in, picking up, or requesting delivery
• Manage Customers: Keep track of customer information and order history
• Menu Management: Create and update your menu items with prices, descriptions, and nutritional info
• Inventory Control: Track ingredients and manage stock levels
• Process Payments: Handle different payment methods and track revenue
• Run Promotions: Create discount codes and special offers
• Collect Feedback: Gather customer reviews and analyze satisfaction

Who Should Use This Manual?

• Restaurant owners and managers
• Software developers integrating with our API
• Anyone setting up online ordering for their food business

GETTING STARTED - QUICK SETUP GUIDE

What You'll Need Before Starting

Before you begin, make sure you have:
• A computer with internet access
• Python 3.10 or newer installed (Download Python here: https://python.org/downloads/)
• Basic familiarity with using a terminal/command prompt

Step 1: Download the API

First, download the restaurant API to your computer:

git clone https://github.com/B1naryB0t/SWE-Final-Project-Snack-Overflow.git
cd SWE-Final-Project-Snack-Overflow

Don't have Git? You can download the files directly from GitHub by clicking the green "Code" button and selecting "Download ZIP".

Step 2: Set Up Your Environment

Create a safe, isolated environment for the API:

On Windows:
python -m venv .venv
.venv\Scripts\activate

On Mac/Linux:
python -m venv .venv
source .venv/bin/activate

Step 3: Install Required Components

Install all the necessary components:

pip install -r requirements.txt

Step 4: Choose Your Database

The API works with two database options:

Option A: SQLite (Recommended for Beginners)
This is already set up for you! SQLite is perfect for testing and small restaurants. No additional setup required.

Option B: MySQL (For Larger Operations)
If you need a more robust database, edit the file api/dependencies/config.py:

Change this line:
DATABASE_URL = "sqlite:///app.db"

To this (replace with your MySQL details):
DATABASE_URL = "mysql+pymysql://username:password@localhost/restaurant_db"

Step 5: Start Your Restaurant API

Launch the API server:

uvicorn api.main:app --reload

Success! You should see a message like:
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)

Step 6: Explore Your API

Open your web browser and visit:
• API Documentation: http://127.0.0.1:8000/docs

HOW TO USE THE API - STEP BY STEP EXAMPLES

Understanding API Basics

The API uses standard web requests:

• GET: Retrieve information (like viewing menu items)

• POST: Create new items (like adding a customer)

• PUT: Update existing items (like changing a menu price)

• DELETE: Remove items (like deleting an old promotion)

Example 1: Setting Up Your Menu

Let's start by adding items to your restaurant menu.

Adding a Menu Item

What you're doing: Creating a new menu item

API Endpoint: POST /menu_item

Information needed:

```
{
  "name": "Burger",
  "category": "Main Course",
  "price": 12.99,
  "calories": 650,
  "tags": "spicy"
}
```

What happens: The API creates your menu item and gives it a unique ID number.

Response you'll get:


```
{
  "name": "Burger",
  "category": "Main Course",
  "price": 12.99,
  "calories": 650,
  "tags": “spicy”,
  "id": 7
}
```


Viewing Your Menu

What you're doing: Looking at all menu items

API Endpoint: GET /menu_item

Information needed: None

Response you'll get: A list of all your menu items

Example 2: Managing Customers

Adding a New Customer

What you're doing: Registering a new customer

API Endpoint: POST /customer

Information needed:

```
{
  "name": "Sarah Johnson",
  "email": "sarah@email.com",
  "phone": "123-456-7890"
}
```

Response you'll get:

```
{
  "name": "Sarah Johnson",
  "email": "sarah@email.com",
  "phone": "123-456-7890",
  "id": 7
}
```

Example 3: Taking Orders

Creating a Customer Order

What you're doing: Processing a new food order

API Endpoint: POST /order

Information needed:
Request Body:

```
{
  "date": "2024-01-15T12:30:00",
  "status": "pending",
  "order_type": "dine-in",
  "customer_id": 1,
  "guest_name": null,
  "guest_email": null,
  "guest_phone": null,
  "promotion_id": null,
  "payments": [
    {
      "status": "completed",
      "type": "credit_card",
      "transaction_id": "txn_123456789",
      "total": 25.98
    }
  ],
  "menu_item_ids": [1, 2]
}
```

What this means:

- Customer #1 placed an order
- They want menu items #1 and #2

• They're dining in the restaurant
• They paid $25.98 with a credit card
• The order is pending (being prepared)

Creating a Guest Order (No Account Required)

What you're doing: Taking an order from someone without an account

Information needed:


```
{
  "date": "2025-08-07T23:34:34.791Z",
  "status": "pending",
  "order_type": "dine-in",
  "customer_id": null,
  "guest_name": "Sarah",
  "guest_email": "Johnson",
  "guest_phone": "1234567890",
  "promotion_id": 0,
  "payments": [
     "status": "completed",
     "type": "credit_card",
     "transaction_id": "txn_123456789",
     "total": 25.98

],
  "menu_item_ids": [1,2]
}
```

Example 4: Managing Inventory

Adding Ingredients

What you're doing: Adding ingredients to your inventory

API Endpoint: POST /ingredient

Information needed:

```
{
  "name": "Ground Beef",
  "quantity": 50,
  "unit": "pounds",
}
```

Checking Ingredient Levels

What you're doing: Seeing what ingredients you have

API Endpoint: GET /ingredient

Information needed: None

Example 5: Creating Promotions

Setting Up a Discount

What you're doing: Creating a promotional discount

API Endpoint: POST /promotion

Information needed:


```
{
  "code": "HAPPY25",
  "discount_amount": 25,
  "expiration_date": "2025-08-013T23:38:03.888Z"
}
```


Example 6: Viewing Reports

Checking Revenue

What you're doing: Seeing how much money you've made

API Endpoint: GET /payment/revenue/

Information needed: Date range
• start_date: 2025-08-01
• end_date: 2025-08-31

Full URL: GET /payment/revenue/?start_date=2025-08-01&end_date=2025-08-31

Response you'll get:


```
{
  "total_revenue": 42.5,
  "total_orders": 4,
  "total_customers": 4
}
```


Finding Your Best Customers

What you're doing: Seeing who spends the most

API Endpoint: GET /analytics/top-customers

Information needed: None

Response you'll get:


```
[
  {
    "name": "Sarah Johnson",
    "email": "sarah@email.com",
```

```
    "total_spent": 156.75
  },
  {
    "name": "Mike Wilson",
    "email": "mike@email.com",
    "total_spent": 134.50
  }
]
```


COMMON TASKS AND HOW-TO GUIDES

How to Track an Order

Customer Question: "Where is my order?"

What to do: Use the tracking number

API Endpoint: GET /order/track/{tracking_number}

Example: GET /order/track/1001

How to Update Menu Prices

What you're doing: Changing the price of an existing menu item

API Endpoint: PUT /menu_item/{menu_item_id}

Example: PUT /menu_item/1

Information needed:
Menu_item_id and your updated item:


```
{
  "name": "Burger",
  "category": "Entree",
  "price": 12.99,
  "calories": 600,
  "tags": "spicy"
}
```


How to Handle Refunds

What you're doing: Processing a payment refund

API Endpoint: PUT /payment/{payment_id}

Information needed:
Payment_id (This is the id number associated with your payments response when you created the order) and updated values


```
{
  "status": "refunded",
  "type": "canceled",
  "transaction_id": "txn678696",
  "total": 12.99,
  "order_id": 1,
}
```


How to Temporarily Disable a Menu Item

What you're doing: Removing an item from the menu temporarily

API Endpoint: DELETE /menu_item/{menu_item_id}

Note: This removes the item completely. To bring it back, you'll need to create it again.

UNDERSTANDING RESPONSES AND ERROR MESSAGES

Successful Responses

When everything works correctly, you'll receive:

• 200 OK: Request successful, data returned
• 201 Created: New item created successfully
• 204 No Content: Action completed, nothing to return (like deletions)

Common Error Messages

• 400 Bad Request: Check your data format
• 404 Not Found: The item you're looking for doesn't exist
• 422 Validation Error: Required information is missing or incorrect

Example Error Response:

```
{
  "detail": [
    {
      "loc": ["body", "email"],
      "msg": "field required",
      "type": "value_error.missing"
    }
  ]
}
```

What this means: You forgot to include an email address in your request.

ADVANCED FEATURES

Guest vs. Registered Customer Orders

Guest Orders: Perfect for walk-in customers
• Include guest_name, guest_email, guest_phone
• Leave customer_id empty, type in null

Registered Customer Orders: For repeat customers
• Include customer_id
• Leave guest information empty

Order Types

Choose the right order type for your situation:
• "dine-in": Customer eating at restaurant
• "takeout": Customer picking up order
• "delivery": Order delivered to customer

Payment Types

Support multiple payment methods:
• "cash": Cash payments
• "credit_card": Credit card payments
• "debit_card": Debit card payments
• "mobile_payment": Apple Pay, Google Pay, etc.

TIPS FOR SUCCESS

Best Practices

1. Keep Menu Updated: Regularly update prices and availability
2. Monitor Inventory: Check ingredient levels daily
3. Review Analytics: Use the analytics endpoints to understand your business
4. Handle Errors Gracefully: Always check response codes
5. Backup Data: Regular database backups are essential

Performance Tips

1. Use Specific Queries: Instead of getting all orders, filter by date range
2. Limit Large Requests: Break up large operations into smaller ones
3. Monitor Response Times: If requests are slow, consider database optimization

Troubleshooting Common Issues

Problem: "I can't connect to the API"
Solution: Make sure the server is running and you're using the correct URL

Problem: "My menu items aren't showing up"
Solution: Check that you created them with POST /menu_item

Problem: "Orders aren't calculating totals correctly"
Solution: Verify that your menu item prices are set correctly

GETTING HELP AND SUPPORT

Self-Help Resources

1. API Documentation: Visit http://127.0.0.1:8000/docs while your server is running
2. Check Logs: Look at your terminal/command prompt for error messages
3. Verify Data: Double-check that all required fields are included

Need More Help?

If you're still having trouble:

1. Check the GitHub repository for updates
2. Review the installation steps to ensure everything was set up correctly
3. Contact the development team through the project repository

QUICK REFERENCE - ALL AVAILABLE ENDPOINTS

Customers

• GET /customer - View all customers
• POST /customer - Add new customer
• GET /customer/{id} - View specific customer
• PUT /customer/{id} - Update customer
• DELETE /customer/{id} - Remove customer

Menu Items

• GET /menu_item - View all menu items
• POST /menu_item - Add new menu item
• GET /menu_item/{id} - View specific menu item
• PUT /menu_item/{id} - Update menu item
• DELETE /menu_item/{id} - Remove menu item
• GET /menu_item/search?q={query} - Search menu items
• GET /menu_item/by_category/{category} - Filter by category

Orders

• GET /order - View all orders
• POST /order - Create new order
• GET /order/{id} - View specific order
• PUT /order/{id} - Update order
• DELETE /order/{id} - Cancel order
• GET /order/track/{tracking_number} - Track order
• GET /order/by_customer/{customer_id} - Customer's orders

Payments & Revenue

• GET /payment - View all payments
• POST /payment - Record new payment
• GET /payment/revenue/ - Revenue report

Analytics

• GET /analytics/top-customers - Best customers by spending

And More!

Visit http://127.0.0.1:8000/docs for the complete list of all available endpoints.

Congratulations! You're now ready to manage your restaurant with the Snack Overflow API. Start with adding a few menu items and taking your first order. Remember, the interactive documentation at /docs is always available to help you explore and test the API.

Happy restaurant managing!