API DOCUMENTATION

FOOD DELIVERY V1

Done By: Daniel Majil

Teacher: Mr. Lester Pech
University of Belize
CMPS4191: Advanced Web-Technologies
October 24,2023

Introduction

Welcome to the Food Delivery API documentation. This guide will help you understand how to use our API to manage customers, food items, orders, and order items. We've provided detailed information on endpoints, request examples, and response formats to make your integration process smooth. This API empowers developers to perform various actions within the food delivery ecosystem, including functions like placing orders, updating order status, adding new menu items, and managing user profiles. Its central purpose is to provide developers with a comprehensive understanding of request formats, enabling them to execute operations that result in the desired outcomes across the food delivery platform. Additionally, any encountered issues or errors are promptly relayed to developers, streamlining the troubleshooting process. This API is tailored for developers who possess expertise in working with RESTful APIs, web development, and databases.

How to Start

To set up the database for the food delivery application, you'll want to use the provided SQL dump file, "food_delivery_sql-dump.sql," located in the "dbfiles" directory. First, create a new database named "FOODAPP" Then, execute the contents of the "food_delivery_sql-dump.sql" file within this newly created database.

Make sure to access the necessary login credentials from the "handler.php" file to configure the connection to your MySQL database. If you're using Apache as your web server, it's recommended to set up the root directory to be the parent directory of the repository, alongside the port which is:

127.0.0.1:8036/CMPS4191_foodapp_apiI/…

Endpoints

Customers

1. Get a List of Customers

Endpoint: /customers

Method: GET

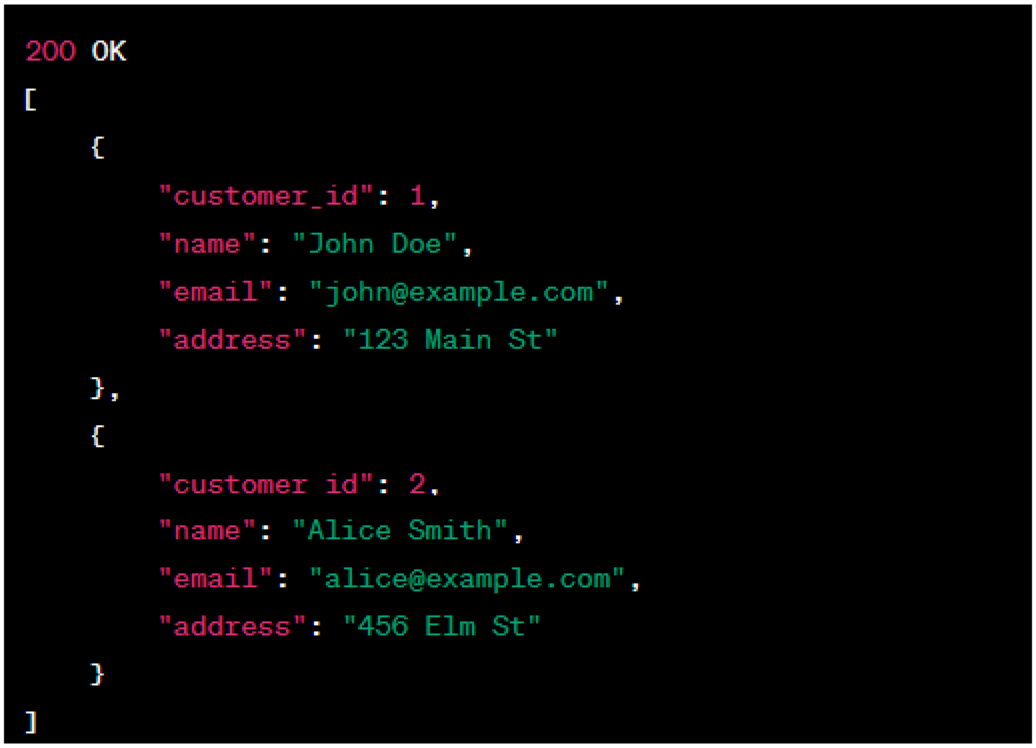
Description: Retrieve a list of all customers.

Parameters: None

Request Example:

```
GET /customers
```

Response Example:



```
200 OK
[
    {
        "customer_id": 1,
        "name": "John Doe",
        "email": "john@example.com",
        "address": "123 Main St"
    },
    {
        "customer id": 2,
        "name": "Alice Smith",
        "email": "alice@example.com",
        "address": "456 Elm St"
    }
]
```

2. Get Customer by ID

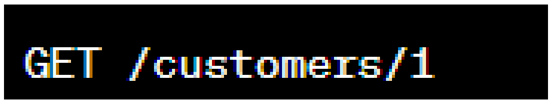
Endpoint: /customers/{customer_id}

Method: GET

Description: Retrieve a customer's details by ID.

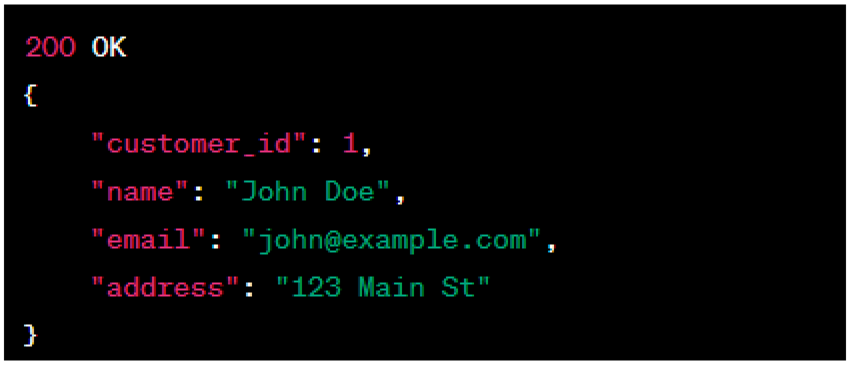
Parameters: {customer_id} - Customer ID

Request Example:



```
GET /customers/1
```

Response Example:



```
200 OK
{
    "customer_id": 1,
    "name": "John Doe",
    "email": "john@example.com",
    "address": "123 Main St"
}
```

3. Create a New Customer

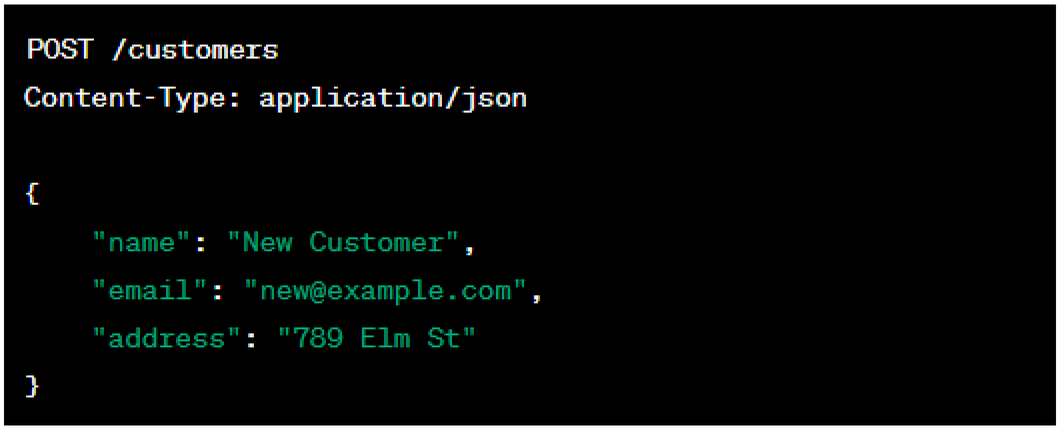
Endpoint: /customers

Method: POST

Description: Create a new customer.

Parameters: JSON data including name, email, and address.

Request Example:



```
POST /customers
Content-Type: application/json

{
    "name": "New Customer",
    "email": "new@example.com",
    "address": "789 Elm St"
}
```

Response Example:

```
201 Created
{
    "customer_id": 3,
    "name": "New Customer",
    "email": "new@example.com",
    "address": "789 Elm St"
}
```

Food Items

4. Get a List of Food Items

Endpoint: /food_items

Method: GET

Description: Retrieve a list of all food items.

Parameters: None

Request Example:

```
GET /food_items
```

Response Example:

```
200 OK
[
    {
        "item_id": 1,
        "name": "Pizza",
        "price": 12.99
    },
    {
        "item_id": 2,
        "name": "Burger",
        "price": 8.99
    }
]
```

5. Get Food Item by ID

Endpoint: /food_items/{item_id}

Method: GET

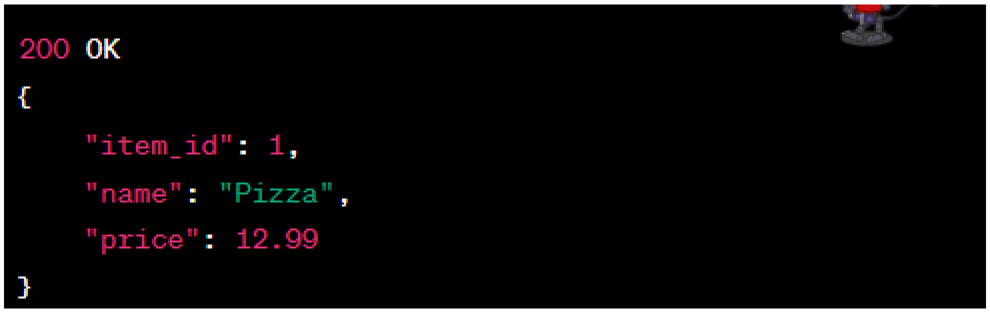
Description: Retrieve a food item's details by ID.

Parameters: {item_id} - Food Item ID

Request Example:

```
GET /food_items/1
```

Response Example:



```
200 OK
{
    "item_id": 1,
    "name": "Pizza",
    "price": 12.99
}
```

6. Create a New Food Item

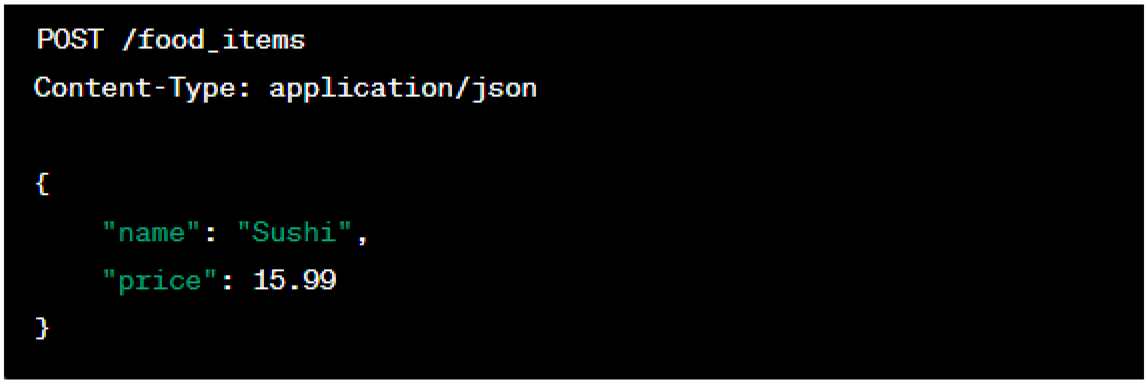
Endpoint: /food_items

Method: POST

Description: Create a new food item.

Parameters: JSON data including name and price.

Request Example:



```
POST /food_items
Content-Type: application/json

{
    "name": "Sushi",
    "price": 15.99
}
```

Response Example:

```
201 Created
{
    "item_id": 3,
    "name": "Sushi",
    "price": 15.99
}
```

Orders

7. Get a List of Orders

Endpoint: /orders

Method: GET

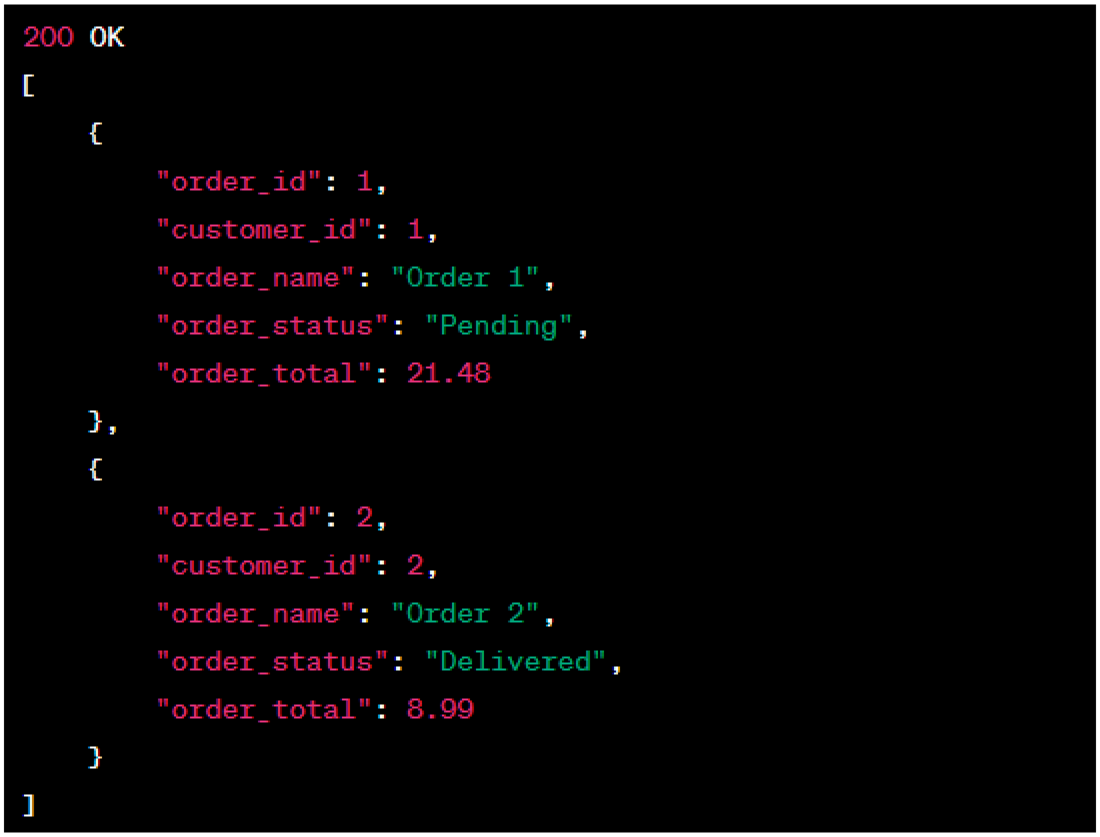
Description: Retrieve a list of all orders.

Parameters: None

Request Example:

```
GET /orders
```

Response Example:



```
200 OK
[
    {
        "order_id": 1,
        "customer_id": 1,
        "order_name": "Order 1",
        "order_status": "Pending",
        "order_total": 21.48
    },
    {
        "order_id": 2,
        "customer_id": 2,
        "order_name": "Order 2",
        "order_status": "Delivered",
        "order_total": 8.99
    }
]
```

8. Get Order by ID

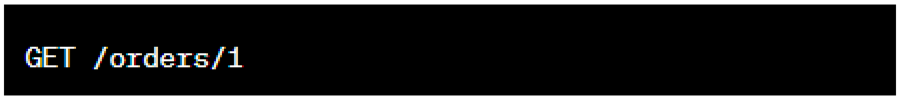
Endpoint: /orders/{order_id}

Method: GET

Description: Retrieve an order's details by ID.

Parameters: {order_id} - Order ID

Request Example:



```
GET /orders/1
```

Response Example:

```
200 OK
{
    "order_id": 1,
    "customer_id": 1,
    "order_name": "Order 1",
    "order_status": "Pending",
    "order_total": 21.48
}
```

9. Create a New Order

Endpoint: /orders

Method: POST

Description: Create a new order.

Parameters: JSON data including customer_id, order_name, and order_total.

Request Example:

```
POST /orders
Content-Type: application/json

{
    "customer_id": 2,
    "order_name": "Order 3",
    "order_total": 12.99
}
```

Response Example:

```
201 Created
{
    "order_id": 3,
    "customer_id": 2,
    "order_name": "Order 3",
    "order_status": "Pending",
    "order_total": 12.99
}
```

Order Items

10. Get Order Items for an Order

Endpoint: /orders/{order_id}/items

Method: GET

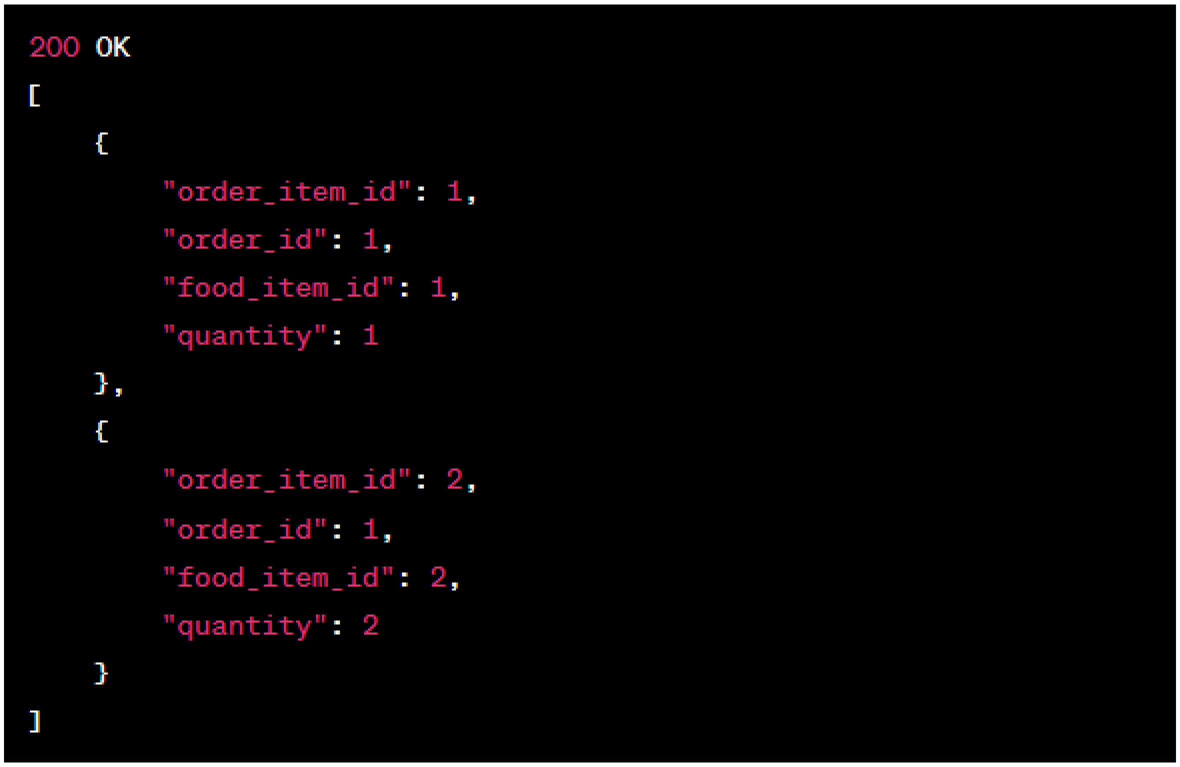
Description: Retrieve the items for a specific order.

Parameters: {order_id} - Order ID

Request Example:

```
GET /orders/1/items
```

Response Example:



```
200 OK
[
    {
        "order_item_id": 1,
        "order_id": 1,
        "food_item_id": 1,
        "quantity": 1
    },
    {
        "order_item_id": 2,
        "order_id": 1,
        "food_item_id": 2,
        "quantity": 2
    }
]
```

11. Add an Item to an Order

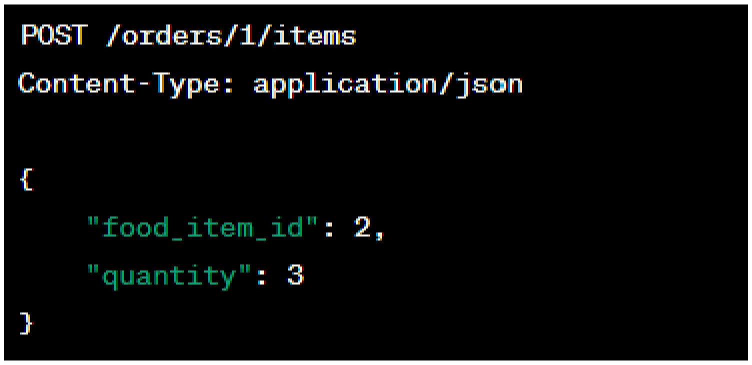
Endpoint: /orders/{order_id}/items

Method: POST

Description: Add a new item to an existing order.

Parameters: {order_id} - Order ID, JSON data including food_item_id and quantity.

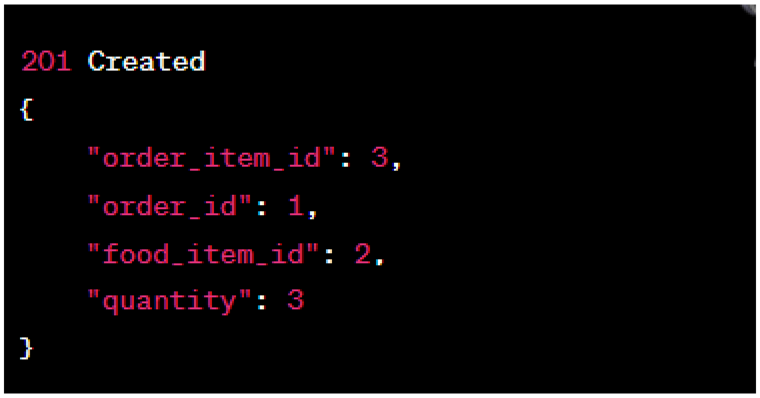
Request Example:



```
POST /orders/1/items
Content-Type: application/json

{
    "food_item_id": 2,
    "quantity": 3
}
```

Response Example:



```
201 Created
{
    "order_item_id": 3,
    "order_id": 1,
    "food_item_id": 2,
    "quantity": 3
}
```

HTTP Status Codes

200 OK: The request was successful, and the response contains the expected data.

201 Created: The resource has been successfully created.

204 No Content: The request was successful, but there is no data to return (e.g., for DELETE requests).

400 Bad Request: The request is malformed or missing required parameters.

401 Unauthorized: Authentication failed, or the provided API key is invalid.

404 Not Found: The requested resource does not exist.

405 Method Not Allowed: The HTTP method used is not supported for the requested endpoint.

500 Internal Server Error: An unexpected server error occurred.

Error Handling

Error Responses

Our API handles errors with informative responses. Here are some common error responses:

400 Bad Request: Invalid request, missing parameters, or incorrect data format.

```
{
  "error": "Invalid request. Please check your parameters."
}
```

401 Unauthorized: Authentication failed or user lacks proper permissions.

```
{
  "error": "Authentication failed. Please check your API key."
}
```

404 Not Found: The requested resource does not exist.

```
{
  "error": "Resource not found. Check the ID or endpoint."
}
```

500 Internal Server Error: An error occurred on our server. Please report it to our support team.

```
{
   "error": "Internal server error. Please contact support."
}
```

Pagination

Pagination of Results

For endpoints that return long lists of resources, you can paginate results using the page and limit query parameters. For example:

GET /food_items?page=2&limit=10

This request would return the second page of results, with each page containing up to 10 items.

Search and Filtering

Query Parameters

Our API allows you to filter and search for resources using query parameters. For example, you can search for customers by name:

GET /customers?name=John Doe

This request would return customers with the name "John Doe."

Versioning

API Versioning

Our API is currently in version 1 (v1). We plan to release updates in the future, and versioning will be handled in the endpoint URLs. For example:

GET /v1/customers

Data Formats and Content Negotiation

Data Formats

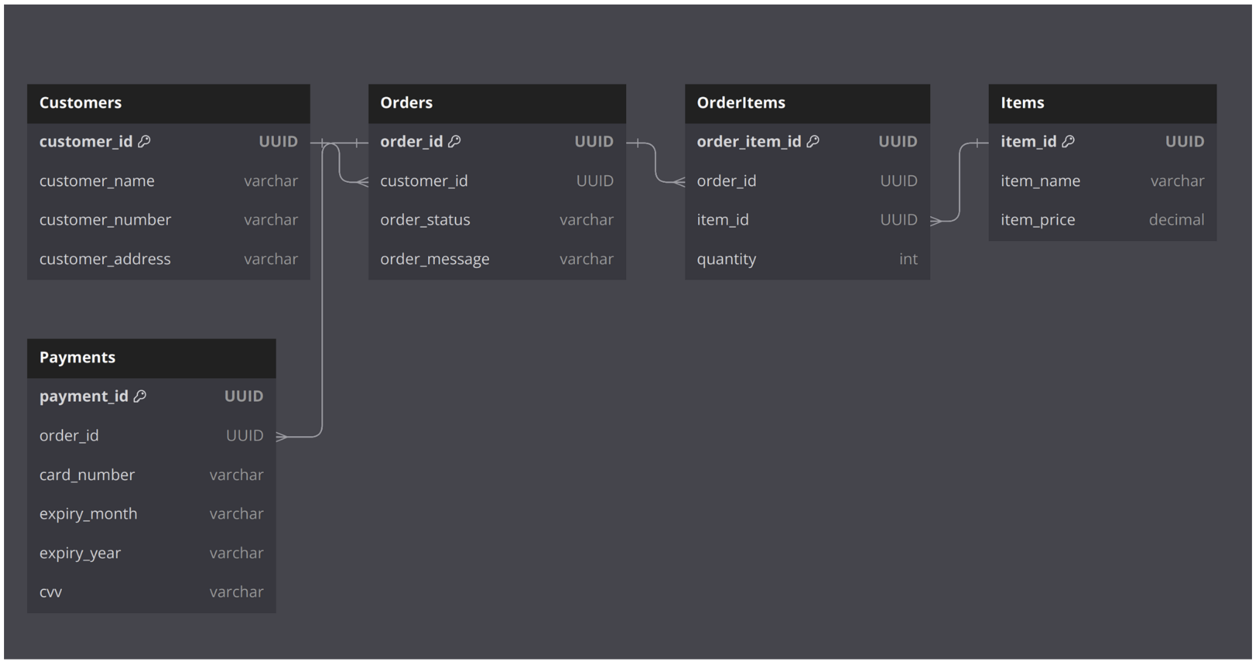
Our API primarily uses JSON for both request and response data.

Content Negotiation

You can specify the desired response format using the Accept header. To request JSON, use:

Accept: application/json

Appendix



Database Relationship Diagram.

Conclusion

This concludes the Food Delivery API Documentation. You now have the knowledge to interact with the API to manage customers, food items, orders, and order items. Make sure to handle different HTTP status codes properly in your implementation. If you encounter any issues or have further questions, please contact our support team at 2020152250@ub.edu.bz.