Restaurant Management API

Technical Documentation

Dcoders

Members: Akhi Chappidi, Dave Nallipogu, Tithi Thakkar, Tyler Webber

Date: August 7, 2025

1. Architecture Overview

The API backend is built using FastAPI and separates concerns by feature. (Customers, menu items, orders, etc.)

- **FastAPI**: Provides routing and access to SwaggerUI docs.
- **SQLAlcehmy**: Connects Python classes to MySQL database.
- **Pydantic**: Validates request and response data.
- **Pytest:** Used for unit tests.

The app runs locally using **Uvicorn** as an ASGI server. No other deployment configured as of now.

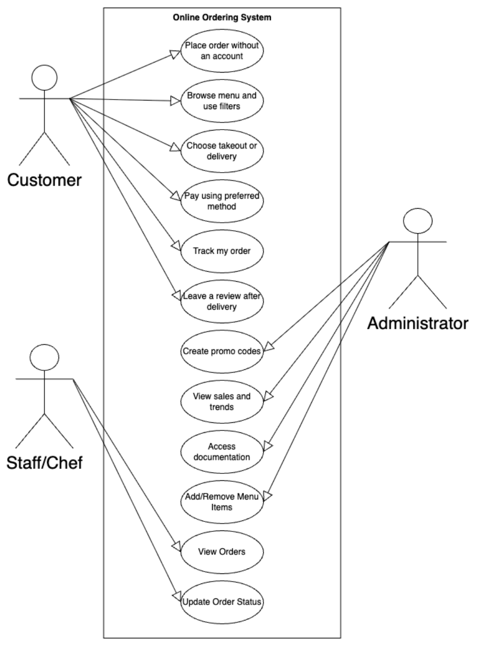
Project Structure

api/

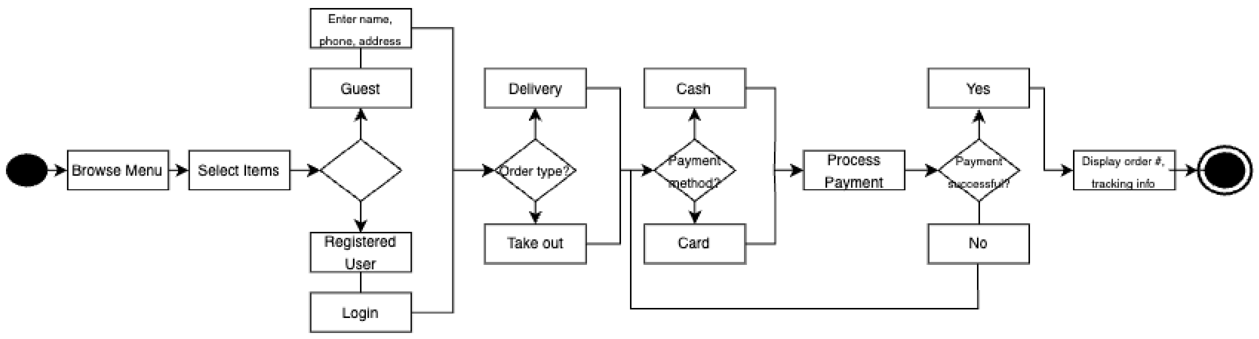
Controllers - Provides business logic such as order management
Dependencies - Contains shared resources
Models - Database models using SQLAlchemy
Routers - API endpoints grouped by feature
Schemas - Data validation with Pydantic models
Tests - Automated tests for features

main.py - Starts the application
Requirements.txt - Contains dependencies

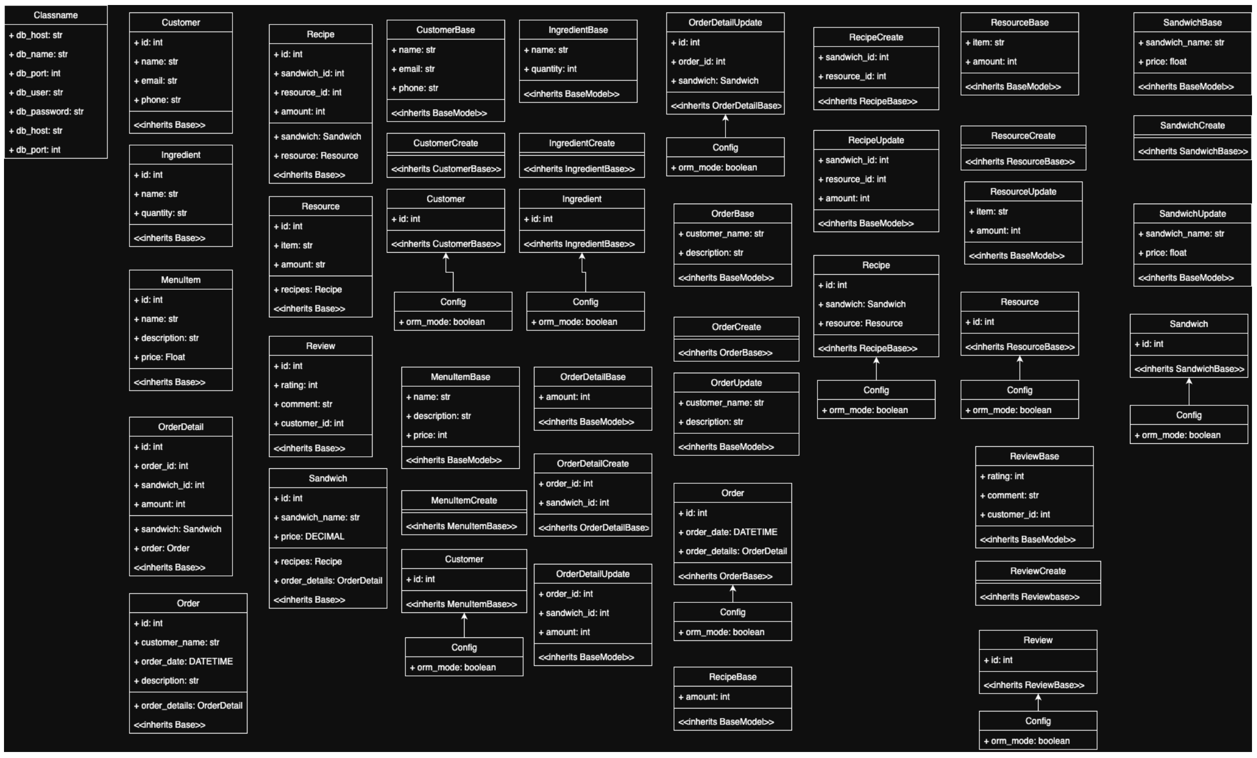
Use Case Diagram:



Activity Diagram:



Class Diagram:



2.1 Endpoint Documentation

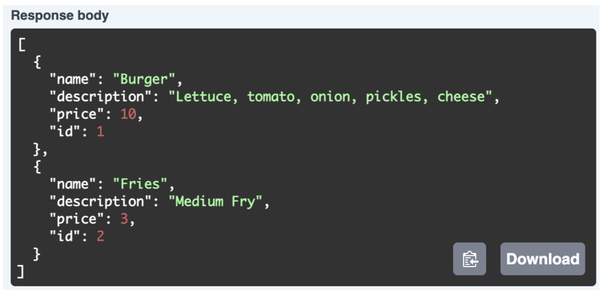
Resource	Method	Endpoint	Description
Orders	GET	/orders/	Retrieve list of all orders.
Orders	POST	/orders/	Create new order,

			updates inventory.
Orders	GET	/orders/{item_id}	Retrieve specific order info by ID.
Orders	PUT	/orders/{item_id}	Update specific order by ID
Orders	DELETE	/orders/{item_id}	Delete specific order by ID.
Orders	GET	/orders/filter	Get orders by date range.
Orders	GET	/orders/revenue	Get total revenue.
Order Details	GET	/orderdetails/	Retrieve all order details.
Order Details	POST	/orderdetails/	Create new order detail entry. (specific item, amount).
Order Details	GET	/orderdetails/{item_id}	Retrieve specific order details by ID.
Order Details	PUT	/orderdetails/{item_id}	Update specific order detail by ID.
Order Details	DELETE	/orderdetails/{item_id}	Delete specific order details by ID.
Customers	POST	/customers/	Create a new customer profile.
Customers	GET	/customers/{customer_id}	Retrieve customer info by customer ID.
Customers	PUT	/customers/{customer_id}	Update customer info by ID.
Customers	DELETE	/customers/{customer_id}	Delete customer info entry.
Menu Items	GET	/menu_items/	Retrieve items on the menu.
Menu Items	POST	/menu_items/	Create an item for the menu.
Menu Items	GET	/menu_items/{item_id}	Retrieve specific menu item info by ID.

Menu Items	PUT	/menu_items/{item_id}	Update specific menu item by ID.
Menu Items	DELETE	/menu_items/{item_id}	Delete menu item by ID.
Ingredients	GET	/ingredients/	Lists all ingredient inventory.
Ingredients	POST	/ingredients/	Create a new entry for an ingredient.
Ingredients	GET	/ingredients/{ingredient_id}	Retrieve specific ingredient info by ID.
Ingredients	PUT	/ingredients/{ingredient_id}	Update specific ingredient by ID.
Ingredients	DELETE	/ingredients/{ingredient_id}	Delete specific ingredient by ID.
Reviews	GET	/reviews/	Retrieve all reviews.
Reviews	POST	/reviews/	Create a new review.
Reviews	GET	/reviews/{review_id}	Retrieve a specific review by ID.
Reviews	PUT	/reviews/{review_id}	Update a specific review by ID.
Reviews	DELETE	/reviews/{review_id}	Delete a specific review by ID.

2.2 Endpoint Response Examples

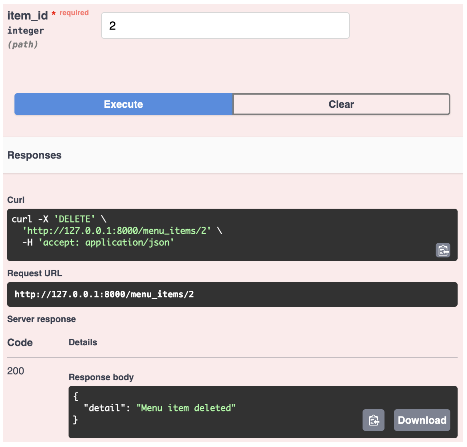
(Get) Menu Items



```
[
  {
    "name": "Burger",
    "description": "Lettuce, tomato, onion, pickles, cheese",
    "price": 10,
    "id": 1
  },
  {
    "name": "Fries",
    "description": "Medium Fry",
    "price": 3,
    "id": 2
  }
]
```

Returns a list of all items on the menu with attributes

(Delete) Menu Item


item_id * required
integer
(path)

```
2
```

Responses

Curl

```
curl -X 'DELETE' \
  'http://127.0.0.1:8000/menu_items/2' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/menu_items/2
```

Server response

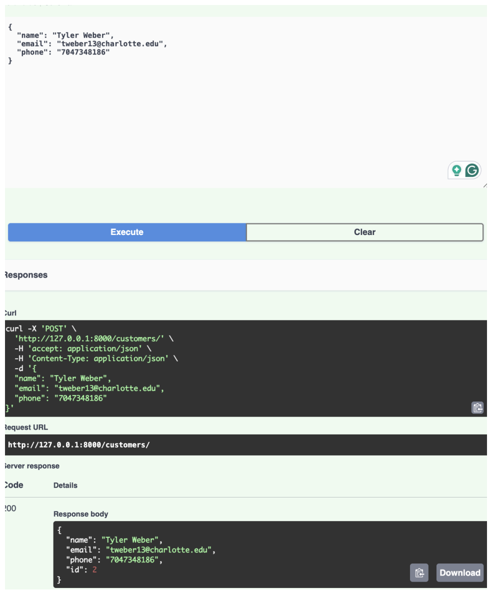
Code Details

200 Response body

```
{
  "detail": "Menu item deleted"
}
```

Deletes an item from the menu based on the item ID given.

(Post) Create Customer



```
{
  "name": "Tyler Weber",
  "email": "tweber13@charlotte.edu",
  "phone": "7047348186"
}
```

esponses

url

```
curl -X 'POST' \
  'http://127.0.0.1:8000/customers/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "name": "Tyler Weber",
  "email": "tweber13@charlotte.edu",
  "phone": "7047348186"
}'
```

equest URL

```
http://127.0.0.1:8000/customers/
```

erver response

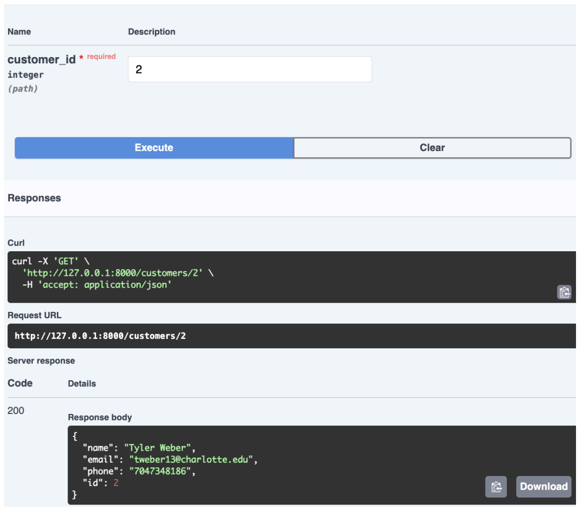
ode Details

00 Response body

```
{
  "name": "Tyler Weber",
  "email": "tweber13@charlotte.edu",
  "phone": "7047348186",
  "id": 2
}
```

Takes a customer's name, email, and phone number and adds them to the database of customers.

(Get) Customer



customer_id * required
integer
(path)

2

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/customers/2' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/customers/2
```

Server response

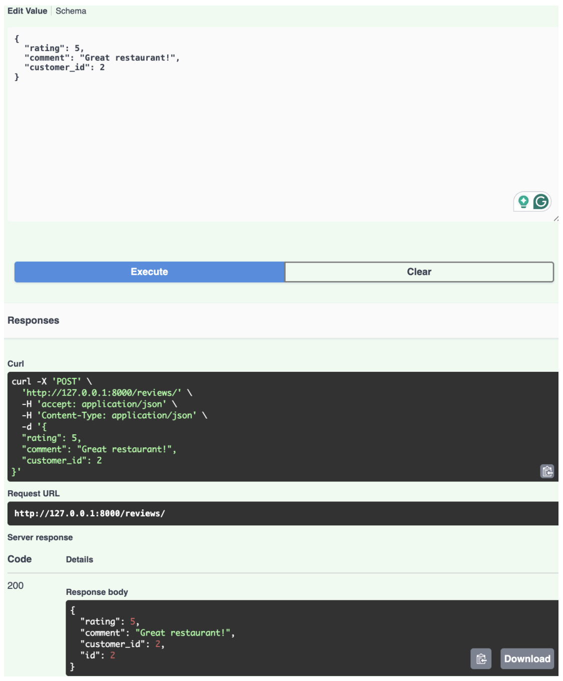
Code Details

200 Response body

```
{
  "name": "Tyler Weber",
  "email": "tweber13@charlotte.edu",
  "phone": "7047348186",
  "id": 2
}
```

Returns the information about a customer based on the ID given.

(Post) Create Review



```
{
  "rating": 5,
  "comment": "Great restaurant!",
  "customer_id": 2
}
```

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/reviews/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "rating": 5,
  "comment": "Great restaurant!",
  "customer_id": 2
}'
```

Request URL

```
http://127.0.0.1:8000/reviews/
```

Server response

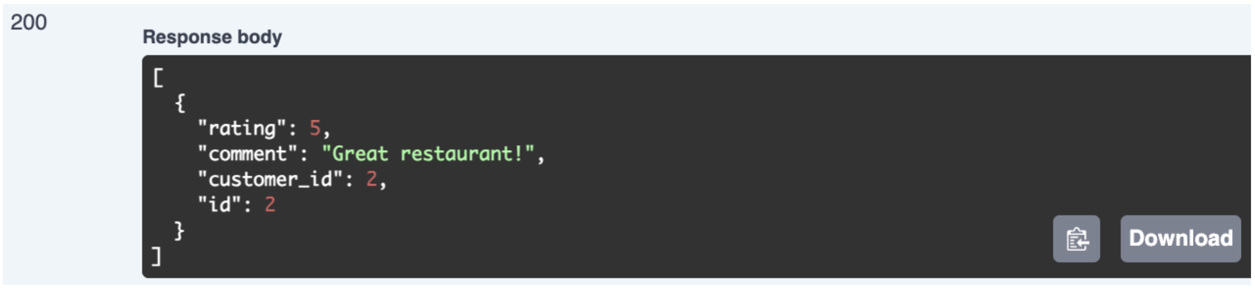
Code Details

200 Response body

```
{
  "rating": 5,
  "comment": "Great restaurant!",
  "customer_id": 2,
  "id": 2
}
```

Allows input of rating, comment, and customer ID, and adds the review to the database. Made for customers to leave reviews of the restaurant.

(Get) Reviews

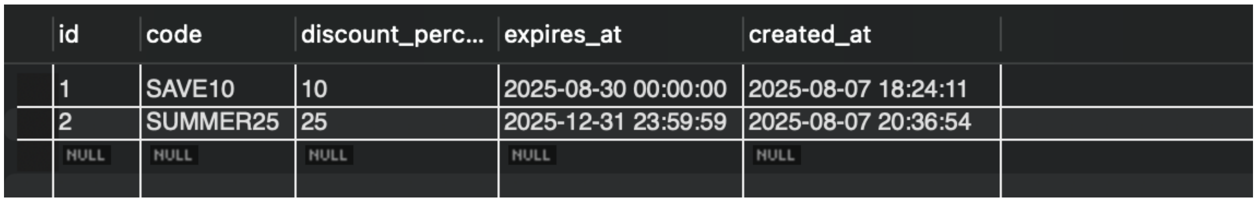

200

Response body

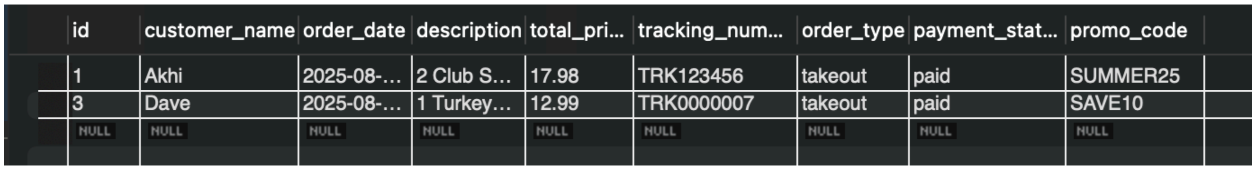
```
[
  {
    "rating": 5,
    "comment": "Great restaurant!",
    "customer_id": 2,
    "id": 2
  }
]
```

Allows for viewing of all reviews, primarily for the restaurant owner to see feedback from customers.

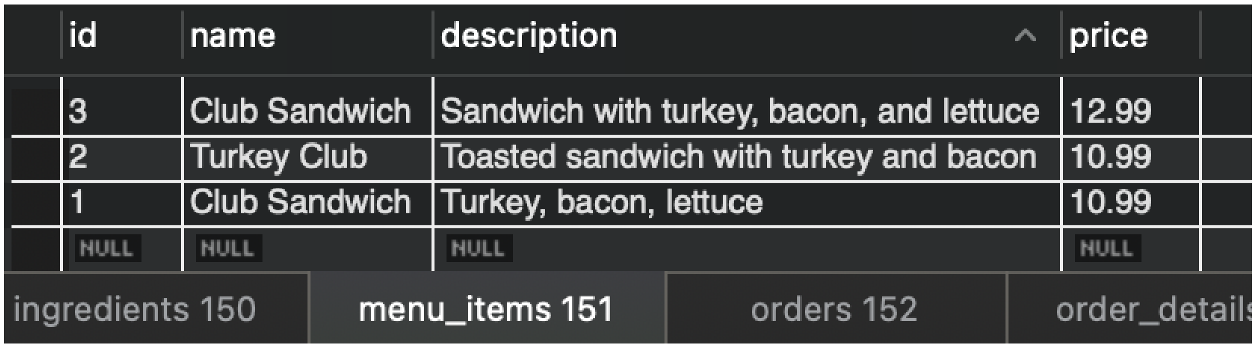
3.1 Database Examples:



id	code	discount_perc...	expires_at	created_at
1	SAVE10	10	2025-08-30 00:00:00	2025-08-07 18:24:11
2	SUMMER25	25	2025-12-31 23:59:59	2025-08-07 20:36:54
NULL	NULL	NULL	NULL	NULL

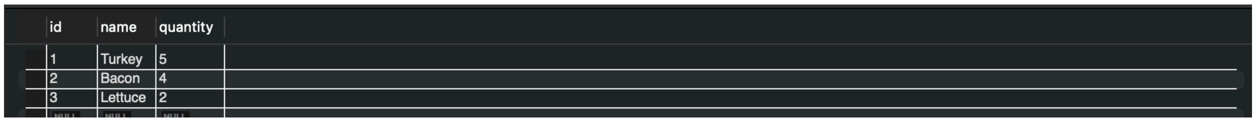


id	customer_name	order_date	description	total_pri...	tracking_num...	order_type	payment_stat...	promo_code
1	Akhi	2025-08-...	2 Club S...	17.98	TRK123456	takeout	paid	SUMMER25
3	Dave	2025-08-...	1 Turkey...	12.99	TRK0000007	takeout	paid	SAVE10
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



id	name	description	price
3	Club Sandwich	Sandwich with turkey, bacon, and lettuce	12.99
2	Turkey Club	Toasted sandwich with turkey and bacon	10.99
1	Club Sandwich	Turkey, bacon, lettuce	10.99
NULL	NULL	NULL	NULL

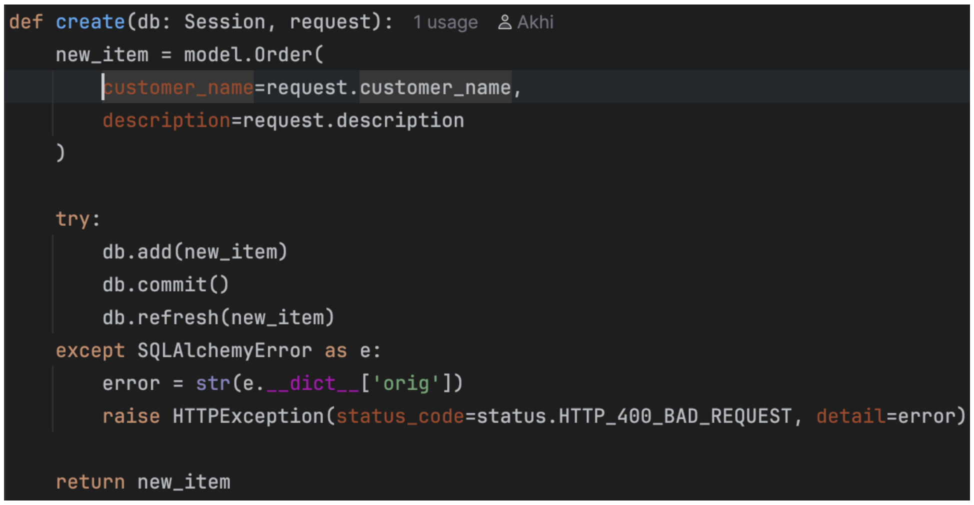
ingredients 150
menu_items 151
orders 152
order_details



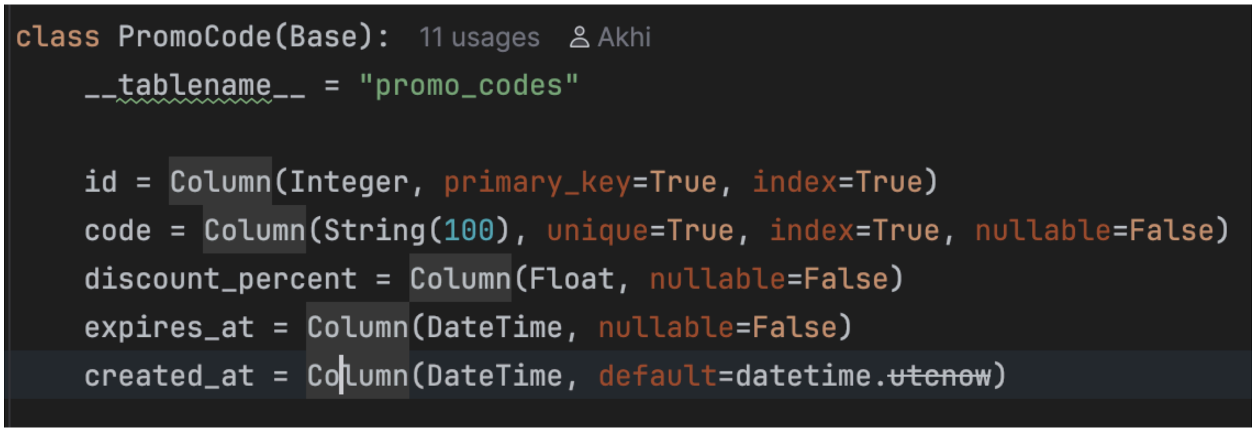
id	name	quantity
1	Turkey	5
2	Bacon	4
3	Lettuce	2

3.2 Code Snippets + Explanations:

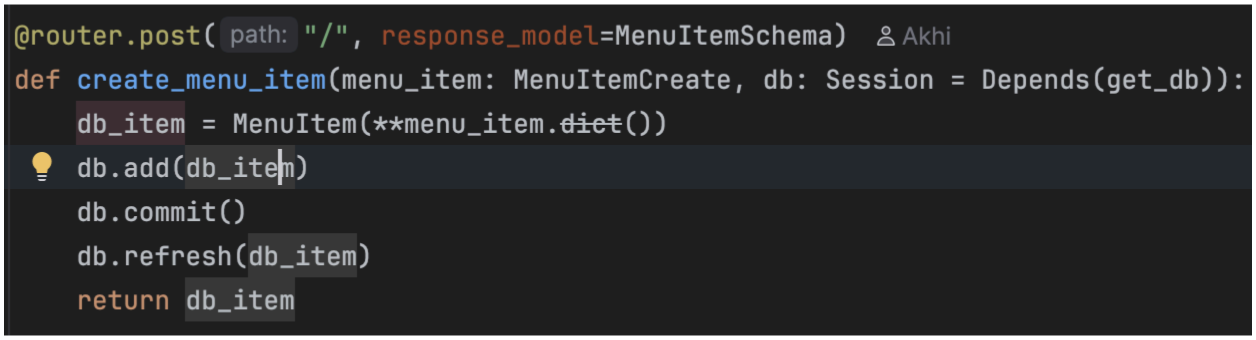
1. Order Controller (api/controllers/orders.py)


```
def create(db: Session, request):  1 usage  Akhi
    new_item = model.Order(
        customer_name=request.customer_name,
        description=request.description
    )

    try:
        db.add(new_item)
        db.commit()
        db.refresh(new_item)
    except SQLAlchemyError as e:
        error = str(e.__dict__['orig'])
        raise HTTPException(status_code=status.HTTP_400_BAD_REQUEST, detail=error)

    return new_item
```
This function creates a new order record in the database using incoming request data.

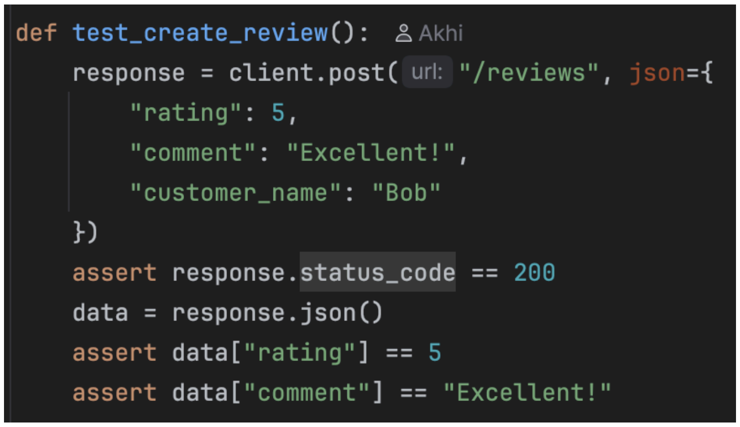
2. PromoCode Model (api/models/promo_code.py)


```
class PromoCode(Base):  11 usages  Akhi
    __tablename__ = "promo_codes"

    id = Column(Integer, primary_key=True, index=True)
    code = Column(String(100), unique=True, index=True, nullable=False)
    discount_percent = Column(Float, nullable=False)
    expires_at = Column(DateTime, nullable=False)
    created_at = Column(DateTime, default=datetime.~~utcnow~~)
```
This SQLAlchemy model maps to the promo_codes table and stores data for each promo code. This includes a unique code, discount percentage, expiration date, and creation timestamp/

3. Menu Item Router (api/routers/menu_item.py)


```
@router.post( path: "/", response_model=MenuItemSchema)  Akhi
def create_menu_item(menu_item: MenuItemCreate, db: Session = Depends(get_db)):
    db_item = MenuItem(**menu_item.~~dict~~())
    db.add(db_item)
    db.commit()
    db.refresh(db_item)
    return db_item
```
Creates a new menu item from request data and stores it in the database. Uses Pydantic schema validation for input and returns the full created item.

4. Review Tests (api/tests/test_reviews.py)



```python
def test_create_review():  Akhi
    response = client.post( url: "/reviews", json={
        "rating": 5,
        "comment": "Excellent!",
        "customer_name": "Bob"
    })
    assert response.status_code == 200
    data = response.json()
    assert data["rating"] == 5
    assert data["comment"] == "Excellent!"
```

This test verifies that creating a review succeeds and returns the expected results.

5. Development Environment Setup:

Prerequisites:

- Python installed
- Git
- Virtual environment tool (ex: Pycharm)

After activating a new virtual environment, for example, in PyCharm,
Go to the terminal and paste

1. git clone https://github.com/akhimass/DCoders-FinalProject.git
2. cd DCoders-FinalProject
3. pip install -r requirements.txt
4. uvicorn main:app –reload

Open http://localhost:8000/docs to explore functionality.