BACKEND DJANGO TEST CASE

Note:

1. Before starting, please create the database first by running the script.
2. ERD (Entity-Relationship Diagram).

Test Cases

1. **API Customer List:**
    - **Format:** customers.json
    - **Requirements:**
        1. User wants to retrieve all customer data.
        2. User wants to retrieve customer data by customer ID (multiple IDs).
        3. User wants to search for customer data by name.
2. **API: Get Product Detail by Product Code:**
    - **Format:** product_detail.json
    - **Requirements:**
        1. User wants to retrieve product details by scanned code.
        2. If the product status is "hold," display a message.
        3. If stock is 0, display a message.
3. **API: Insert Data Sale:**
    - **Format:** insert_data_sale.json
    - **Requirements:**
        1. User wants to insert transaction data with multiple items.
        2. **Logic:**
            - If quantity > product stock, show an alert and fail the insert.
            - Otherwise, insert data into the tables:
                - **Sales:** Insert header data.
                - **Sale_items:** Insert item data.
        3. If the insert is successful:
            - Update product stock: Stock – qty = latest product stock.
            - Update the sales table column: sale_items_total = total items successfully inserted.
        4. If the transaction is done:
            - Show a message with the format:
                - Display all item statuses: Success & Failed Insert, with format: Return_data_after_insert_data_sale.json.
4. **API Paging:**
    - **Format:** paging.json
    - **Requirements:**
        1. User wants to view all transaction data based on:
            - Date period.
            - Total data to show.
            - Keyword (search by transaction code & customer name).
        2. User can slide the page left and right.
5. **API Cart Compare Data:**
    - **Format:** compare_chart.json

BACKEND DJANGO TEST CASE

- **Requirements:**
    1. User wants to see data comparison with a chart:
        - Compare all transaction data by:
            - Dates:
                - Show total transaction price over time based on date.
                - Order data by time in ascending order.
    2. If the date is empty:
        - Get all data.
        - Show total transaction price over time based on date.
        - Order data by time in ascending order.

6. **API Product Popular:**
    - **Format:** product_popular.json
    - **Requirements:**
        1. User wants to see the top 5 most popular products.
        2. Order data by total_price in descending order.