```
============================
 RESTAURANT MANAGEMENT SYSTEM
============================
1. Display Menu
2. Start New Order
3. Add Item to Order
4. Calculate and Finalize Bill
5. Exit
Enter your choice (1-5): 1

--- Current Menu ---
ID  | Item Name          | Price (Rs)
----|--------------------|----------
101 | Burger Deluxe      | Rs250.00
102 | Veggie Wrap        | Rs180.00
201 | Cola               | Rs75.00
301 | Brownie            | Rs110.00
---------------------------------

============================
 RESTAURANT MANAGEMENT SYSTEM
============================
1. Display Menu
2. Start New Order
3. Add Item to Order
4. Calculate and Finalize Bill
5. Exit
Enter your choice (1-5): 2

--- New Order Started ---

============================
 RESTAURANT MANAGEMENT SYSTEM
============================
1. Display Menu
2. Start New Order
3. Add Item to Order
4. Calculate and Finalize Bill
5. Exit
Enter your choice (1-5): 3

--- Current Menu ---
ID  | Item Name          | Price (Rs)
----|--------------------|----------
101 | Burger Deluxe      | Rs250.00
102 | Veggie Wrap        | Rs180.00
201 | Cola               | Rs75.00
301 | Brownie            | Rs110.00
---------------------------------
Enter Item ID to add: 102
Enter Quantity: 1000
Added 1000 x Veggie Wrap to the order.

============================
 RESTAURANT MANAGEMENT SYSTEM
============================
1. Display Menu
2. Start New Order
3. Add Item to Order
4. Calculate and Finalize Bill
5. Exit
Enter your choice (1-5): 4

--- Final Bill ---
Qty | Item Name          | Unit Price | Line Total
----|--------------------|------------|-----------
1000 | Veggie Wrap       | Rs180.00   | Rs180000.00
-----------------------------------------
                    Subtotal: Rs180000.00
                     Tax (5%): Rs9000.00
                GRAND TOTAL: Rs189000.00
-----------------------------------------
```

```python
import json
import os
from datetime import datetime


MENU_FILE = 'menu_v2.json'
TAX_RATE = 0.05


class MenuItem:
    """Represents a single item on the menu."""
    def __init__(self, item_id, name, price):
        self.item_id = str(item_id)
        self.name = name
        self.price = float(price)

    def __str__(self):
        return f"{self.item_id:<3} | {self.name:<18} | Rs{self.price:.2f}"


class Order:
    """Manages items added to a single customer's bill."""
    def __init__(self):

        self.items = {}

    def add_item(self, item_id, quantity):
        """Adds or updates an item in the order."""
        try:
            quantity = int(quantity)
            if quantity <= 0:
                print("Quantity must be positive.")
                return False

            self.items[item_id] = self.items.get(item_id, 0) + quantity
            return True
        except ValueError:
            print("Invalid quantity. Please enter a number.")
            return False

    def get_order_summary(self, menu_dict):
        """Calculates totals and returns a detailed summary."""
        subtotal = 0.0
        details = []

        for item_id, quantity in self.items.items():
            if item_id in menu_dict:
                item = menu_dict[item_id]
                line_total = item.price * quantity
                subtotal += line_total
                details.append({
                    "name": item.name,
                    "quantity": quantity,
                    "unit_price": item.price,
                    "line_total": line_total
                })

        tax_amount = subtotal * TAX_RATE
        grand_total = subtotal + tax_amount

        return {
            "details": details,
            "subtotal": subtotal,
            "tax": tax_amount,
            "total": grand_total
        }
```

```
        menu = {
            item_id: MenuItem(item_id, data['name'], data['price'])
            for item_id, data in default_data.items()
        }
        return menu

    def _load_menu(self):
        """Loads menu from JSON or creates default menu."""
        if os.path.exists(MENU_FILE):
            try:
                with open(MENU_FILE, 'r') as f:
                    raw_data = json.load(f)


                self.menu_dict = {
                    item_id: MenuItem(item_id, data['name'], data['price'])
                    for item_id, data in raw_data.items()
                }
                print(f"Menu loaded from {MENU_FILE}.")
            except (json.JSONDecodeError, KeyError, FileNotFoundError):
                print("Error loading menu file. Initializing default menu.")
                self.menu_dict = self._initialize_default_menu()
                self._save_menu()
        else:
            print(f"Menu file '{MENU_FILE}' not found. Initializing default menu.")
            self.menu_dict = self._initialize_default_menu()
            self._save_menu()

    def _save_menu(self):
        """Saves the current menu to a JSON file."""

        serializable_data = {
            item.item_id: {"name": item.name, "price": item.price}
            for item in self.menu_dict.values()
        }
        try:
            with open(MENU_FILE, 'w') as f:
                json.dump(serializable_data, f, indent=4)
        except Exception as e:
            print(f"Error saving menu: {e}")

    def _ensure_order_log_file(self):
        """Creates the orders log file if it doesn't exist."""
        if not os.path.exists('orders_log_v2.txt'):
            with open('orders_log_v2.txt', 'w') as f:
                f.write("--- Restaurant Order Log ---\n")


    def display_menu(self):
        """Displays the menu in a formatted table."""
        if not self.menu_dict:
            print("\n--- Menu is Empty ---")
            return

        print("\n--- Current Menu ---")
        print("ID   | Item Name          | Price (Rs)")
        print("-----|--------------------|-----------")


        sorted_keys = sorted(self.menu_dict.keys(), key=lambda x: int(x))

        for item_id in sorted_keys:
            print(self.menu_dict[item_id])
        print("-" * 35)

    def start_order(self):
        """Starts a new Order instance."""
        self.current_order = Order()
        print("\n--- New Order Started ---")

    def add_item_to_order(self, item_id, quantity):
        """Adds an item to the current order after validation."""
        if not self.current_order:
            self.start_order()
```

```
            self.start_order()

        if item_id in self.menu_dict:
            item_name = self.menu_dict[item_id].name
            if self.current_order.add_item(item_id, quantity):
                print(f"Added {quantity} x {item_name} to the order.")
        else:
            print(f"Error: Item ID '{item_id}' not found in the menu.")

    def finalize_bill(self):
        """Calculates, prints, logs the bill, and resets the order."""
        if not self.current_order or not self.current_order.items:
            print("\nCannot finalize: Order is empty.")
            return

        summary = self.current_order.get_order_summary(self.menu_dict)


        print("\n--- Final Bill ---")
        print("Qty | Item Name          | Unit Price | Line Total")
        print("----|--------------------|------------|-----------")
        for detail in summary['details']:
            print(f"{detail['quantity']:<3} | {detail['name']:<18} | Rs{detail['unit_price']:.2f}     | Rs{detail['line_total']:.2f}")

        print("-" * 45)
        print(f"{'Subtotal:':>35} Rs{summary['subtotal']:.2f}")
        print(f"{'Tax (5%):':>35} Rs{summary['tax']:.2f}")
        print(f"{'GRAND TOTAL:':>35} Rs{summary['total']:.2f}")
        print("-" * 45)


        self._log_transaction(summary)


        self.current_order = None
        print("\nOrder finalized and logged. Thank you!")


    def _log_transaction(self, summary):
        """Logs the completed transaction details to a file."""
        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

        log_entry = {
            "timestamp": timestamp,
            "order_details": [{"name": d['name'], "qty": d['quantity'], "price": d['unit_price']} for d in summary['details']],
            "total_amount": summary['total']
        }

        try:
            with open('orders_log_v2.txt', 'a') as f:
                f.write(json.dumps(log_entry) + "\n")
        except Exception as e:
            print(f"Error logging transaction: {e}")


def run_cli():
    """Main loop for the command-line interface."""
    restaurant = Restaurant()

    while True:
        print("\n============================")
        print(" RESTAURANT MANAGEMENT SYSTEM ")
        print("============================")
        print("1. Display Menu")
        print("2. Start New Order")
        print("3. Add Item to Order")
        print("4. Calculate and Finalize Bill")
        print("5. Exit")

        choice = input("Enter your choice (1-5): ").strip()

        if choice == '1':
            restaurant.display_menu()

        elif choice == '2':
            restaurant.start_order()
```

```
        choice = input("Enter your choice (1-5): ").strip()

        if choice == '1':
            restaurant.display_menu()

        elif choice == '2':
            restaurant.start_order()

        elif choice == '3':
            if not restaurant.current_order:
                print("\nNo order started yet. Starting a new one automatically.")
                restaurant.start_order()

            restaurant.display_menu()
            item_id = input("Enter Item ID to add: ").strip()
            quantity = input("Enter Quantity: ").strip()

            if item_id and quantity:
                restaurant.add_item_to_order(item_id, quantity)

        elif choice == '4':
            restaurant.finalize_bill()

        elif choice == '5':
            print("Thank you for using the RMS. Goodbye!")
            break

        else:
            print("Invalid choice. Please enter a number between 1 and 5.")

if __name__ == "__main__":
    run_cli()
```