**Assignment Title: Restaurant Management System**

**Objective**

The goal of this assignment is to build a simple **Restaurant Management System** using Python. This system will allow:

1. Managing the menu (add, remove, or update items).
2. Taking customer orders.
3. Calculating bills with tax and service charges.
4. Handling errors gracefully using exception handling.

**Instructions**

1. **Complete the Program Functionality**:  
   Implement all features listed below.
2. **Use Exception Handling**:  
   Handle errors like invalid input, unavailable items, and improper menu management.

**Requirements**

**Features to Implement**

1. **Menu Management**:
   * Allow the manager to:
     + Add new menu items with prices.
     + Remove items from the menu.
     + Update the price and quantity of existing items.
2. **Take Orders**:
   * Display the menu to customers.
   * Allow customers to select items and quantities.
   * Validate that items exist in the menu.
3. **Bill Calculation**:
   * Calculate the total cost, including:
     + 10% service charge.
     + 5% tax.
4. **Error Handling**:
   * Handle invalid inputs (e.g., non-numeric prices, nonexistent menu items).
5. **Exit System**:
   * Allow users to exit the system gracefully.

**Sample Output:**

**Restaurant System:**

1. Manage Menu
2. Take Order
3. Calculate Bill
4. Exit

Enter your choice:

**Menu Management:**

1. Add Item
2. Remove Item
3. Update Item Price
4. View Menu
5. Exit Management

Enter your choice: 1

Enter item name: Pasta

Enter item price: 12.50

Item added successfully!

**Take Orders**

Menu:

1. Burger - $5.99
2. Pizza - $8.49
3. Salad - $4.99

Enter item name to order: Pizza

Enter quantity: 2

Order added: Pizza x 2

**Bill Calculation**

Order Summary:

Pizza x 2 = $16.98

Tax (5%): $0.85

Service Charge (10%): $1.70

Total: $19.53

**Solution:**

class Restaurant:

def \_\_init\_\_(self):

self.menu = {} # Dictionary to store menu items and prices

self.order = {} # Dictionary to store orders

def manage\_menu(self):

while True:

print("\nMenu Management:")

print("1. Add Item")

print("2. Remove Item")

print("3. Update Item Price")

print("4. View Menu")

print("5. Exit Management")

try:

choice = int(input("Enter your choice: "))

if choice == 1: # Add Item

item\_name = input("Enter item name: ").strip().title()

if item\_name in self.menu:

print("Item already exists in the menu!")

continue

price = float(input("Enter item price: "))

self.menu[item\_name] = price

print(f"Item '{item\_name}' added successfully!")

elif choice == 2: # Remove Item

item\_name = input("Enter item name to remove: ").strip().title()

if item\_name in self.menu:

del self.menu[item\_name]

print(f"Item '{item\_name}' removed successfully!")

else:

print(f"Error: Item '{item\_name}' not found in the menu.")

elif choice == 3: # Update Item Price

item\_name = input("Enter item name to update: ").strip().title()

if item\_name in self.menu:

price = float(input(f"Enter new price for '{item\_name}': "))

self.menu[item\_name] = price

print(f"Price of '{item\_name}' updated successfully!")

else:

print(f"Error: Item '{item\_name}' not found in the menu.")

elif choice == 4: # View Menu

if not self.menu:

print("The menu is empty!")

else:

print("\nCurrent Menu:")

for item, price in self.menu.items():

print(f"{item}: ${price:.2f}")

elif choice == 5: # Exit Management

break

else:

print("Invalid choice. Please try again.")

except ValueError:

print("Error: Please enter a valid number.")

def take\_order(self):

if not self.menu:

print("The menu is empty. Please add items before taking orders.")

return

print("\nCurrent Menu:")

for item, price in self.menu.items():

print(f"{item}: ${price:.2f}")

while True:

try:

item\_name = input("Enter item name to order (or 'done' to finish): ").strip().title()

if item\_name.lower() == 'done':

break

if item\_name not in self.menu:

print(f"Error: '{item\_name}' is not available on the menu.")

continue

quantity = int(input(f"Enter quantity for '{item\_name}': "))

if quantity <= 0:

print("Error: Quantity must be greater than 0.")

continue

# Add the order to the order dictionary

if item\_name in self.order:

self.order[item\_name] += quantity

else:

self.order[item\_name] = quantity

print(f"'{item\_name}' x {quantity} added to the order.")

except ValueError:

print("Error: Please enter a valid number for quantity.")

def calculate\_bill(self):

if not self.order:

print("No items in the order. Please take an order first.")

return

print("\nOrder Summary:")

subtotal = 0

for item, quantity in self.order.items():

price = self.menu[item]

item\_total = price \* quantity

subtotal += item\_total

print(f"{item} x {quantity} = ${item\_total:.2f}")

# Calculate tax and service charge

tax = subtotal \* 0.05

service\_charge = subtotal \* 0.10

total = subtotal + tax + service\_charge

print(f"\nSubtotal: ${subtotal:.2f}")

print(f"Tax (5%): ${tax:.2f}")

print(f"Service Charge (10%): ${service\_charge:.2f}")

print(f"Total: ${total:.2f}")

# Clear the order after calculating the bill

self.order.clear()

def run(self):

while True:

print("\nRestaurant System:")

print("1. Manage Menu")

print("2. Take Order")

print("3. Calculate Bill")

print("4. Exit")

try:

choice = int(input("Enter your choice: "))

if choice == 1:

self.manage\_menu()

elif choice == 2:

self.take\_order()

elif choice == 3:

self.calculate\_bill()

elif choice == 4:

print("Exiting... Thank you!")

break

else:

print("Invalid choice. Please try again.")

except ValueError:

print("Error: Please enter a valid number.")

if \_\_name\_\_ == "\_\_main\_\_":

restaurant = Restaurant()

restaurant.run()