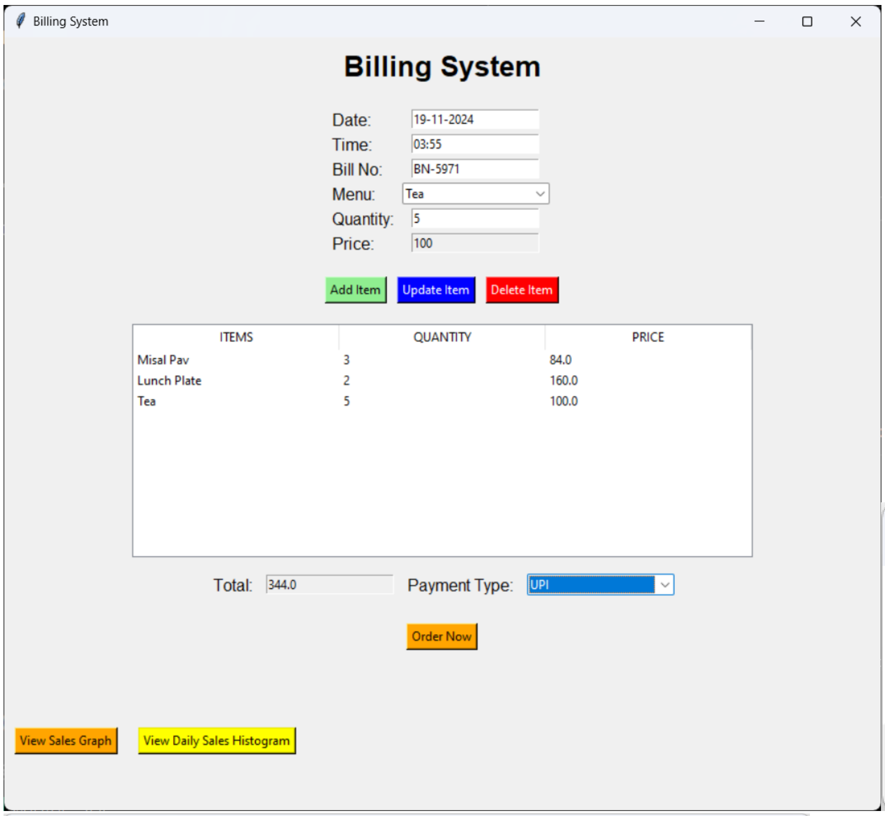
```
import tkinter as tk
from tkinter import ttk, messagebox
from datetime import datetime, timedelta
import random
import sqlite3
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

# Initialize the Tk root window first
root = tk.Tk()
root.title("Billing System")
root.geometry("850x750")

# Initialize database connection
conn = sqlite3.connect('billing_system.db')
cursor = conn.cursor()

# Create table for storing orders if it doesn't exist
cursor.execute('''
    CREATE TABLE IF NOT EXISTS orders (
        bill_number TEXT,
        item TEXT,
        quantity INTEGER,
        price REAL,
        date TEXT,
        time TEXT,
        payment_type TEXT
    )
''')
conn.commit()

# Global variables
order_items = []
menu = {
    "Water Bottle": 20, "Tea": 20, "Coffee": 25,
    "Idli Sambar":22, "Sandwich":40, "Misal Pav":28, 'Noodles':45,
    'Lunch Plate':80
}

# Initialize key variables (after root is created)
bill_number = tk.StringVar(value=f"BN-{random.randint(1000, 9000)}")
date = tk.StringVar(value=datetime.now().strftime('%d-%m-%Y'))
time = tk.StringVar(value=datetime.now().strftime('%H:%M'))
selected_item = tk.StringVar()
quantity = tk.IntVar(value=1)
price = tk.DoubleVar(value=0.0)
total_bill = tk.DoubleVar(value=0.0)
payment_type = tk.StringVar()

# Function Definitions
def update_price(event=None):
    item = selected_item.get()
    qty = quantity.get()
    if item in menu and qty > 0:
        price.set(menu[item] * qty)
```

```
def add_order():
    item = selected_item.get()
    qty = quantity.get()
    prc = price.get()

    if item and qty > 0 and prc > 0:
        order_items.append((item, qty, prc))
        order_list.insert("", "end", values=(item, qty, prc))
        calculate_total()

def delete_order():
    selected = order_list.selection()
    if selected:
        for sel in selected:
            values = order_list.item(sel)["values"]
            item_to_remove = (values[0], int(values[1]), float(values[2]))
            if item_to_remove in order_items:
                order_items.remove(item_to_remove)
            order_list.delete(sel)
        calculate_total()
    else:
        messagebox.showerror("Error", "Please select an item to delete!")

def update_order():
    selected = order_list.selection()
    if selected:
        values = order_list.item(selected[0])["values"]
        item_name = values[0]
        new_qty = quantity.get()
        if new_qty > 0:
            for i, (name, qty, prc) in enumerate(order_items):
                if name == item_name:
                    order_items[i] = (name, new_qty, menu[name] * new_qty)
                    break
            order_list.item(selected[0], values=(item_name, new_qty, menu[item_name] * new_qty))
            calculate_total()
        else:
            messagebox.showerror("Error", "Quantity must be greater than zero!")
    else:
        messagebox.showerror("Error", "Please select an item to update!")

def calculate_total():
    total = sum(item[2] for item in order_items)
    total_bill.set(total)

def order_now():
    if payment_type.get():
        for item, qty, prc in order_items:
            cursor.execute('''
                INSERT INTO orders (bill_number, item, quantity, price, date, time, payment_type)
                VALUES (?, ?, ?, ?, ?, ?, ?)
            ''', (bill_number.get(), item, qty, prc, date.get(), time.get(), payment_type.get()))
        conn.commit()
        messagebox.showinfo("Order Placed", f"Order successful! Total: {total_bill.get()}")
    else:
        messagebox.showerror("Error", "Please select the payment type!")
```

```
def view_sales_graph():
    graph_window = tk.Toplevel(root)
    graph_window.title("Sales Graph")
    graph_window.geometry("800x600")

    cursor.execute('''
        SELECT item, SUM(quantity) FROM orders GROUP BY item
    ''')
    sales_data = cursor.fetchall()

    items = [row[0] for row in sales_data]
    quantities = [row[1] for row in sales_data]

    fig, ax = plt.subplots(figsize=(10, 6))
    ax.bar(items, quantities, color='blue')
    ax.set_xlabel("Items")
    ax.set_ylabel("Quantity Sold")
    ax.set_title("Overall Sales Overview")

    canvas = FigureCanvasTkAgg(fig, master=graph_window)
    canvas.get_tk_widget().pack(pady=20)
    canvas.draw()

def view_daily_sales():
    graph_window = tk.Toplevel(root)
    graph_window.title("Daily Sales Histogram")
    graph_window.geometry("800x600")

    dates = [(datetime.now() - timedelta(days=i)).strftime('%d-%m-%Y') for i in range(7)]
    daily_sales = []
    for day in dates:
        cursor.execute('''
            SELECT SUM(price) FROM orders WHERE date = ?
        ''', (day,))
        result = cursor.fetchone()[0]
        daily_sales.append(result if result else 0)

    fig, ax = plt.subplots(figsize=(10, 6))
    ax.bar(dates, daily_sales, color='green')
    ax.set_xlabel("Date")
    ax.set_ylabel("Sales Amount")
    ax.set_title("Daily Sales for Past Week")

    canvas = FigureCanvasTkAgg(fig, master=graph_window)
    canvas.get_tk_widget().pack(pady=20)
    canvas.draw()

# UI Design
tk.Label(root, text="Billing System", font="Arial 20 bold").pack(pady=10)

frame = tk.Frame(root)
frame.pack(pady=10)

tk.Label(frame, text="Date:", font="Arial 12").grid(row=0, column=0, sticky="w")
tk.Entry(frame, textvariable=date).grid(row=0, column=1, padx=5)
tk.Label(frame, text="Time:", font="Arial 12").grid(row=1, column=0, sticky="w")
tk.Entry(frame, textvariable=time).grid(row=1, column=1, padx=5)

tk.Label(frame, text="Bill No:", font="Arial 12").grid(row=2, column=0, sticky="w")
tk.Entry(frame, textvariable=bill_number).grid(row=2, column=1, padx=5)

tk.Label(frame, text="Menu:", font="Arial 12").grid(row=3, column=0, sticky="w")
menu_combo = ttk.Combobox(frame, textvariable=selected_item, values=list(menu.keys()), state="readonly")
menu_combo.grid(row=3, column=1, padx=5)
menu_combo.bind("<<ComboboxSelected>>", update_price)

tk.Label(frame, text="Quantity:", font="Arial 12").grid(row=4, column=0, sticky="w")
qty_entry = tk.Entry(frame, textvariable=quantity)
qty_entry.grid(row=4, column=1, padx=5)
qty_entry.bind("<KeyRelease>", update_price)

tk.Label(frame, text="Price:", font="Arial 12").grid(row=5, column=0, sticky="w")
tk.Entry(frame, textvariable=price, state="readonly").grid(row=5, column=1, padx=5)

button_frame = tk.Frame(root)
button_frame.pack(pady=10)

tk.Button(button_frame, text="Add Item", bg="light green", command=add_order).pack(side="left", padx=5)
tk.Button(button_frame, text="Update Item", bg="blue", fg="white", command=update_order).pack(side="left",
padx=5)
tk.Button(button_frame, text="Delete Item", bg="red", fg="white", command=delete_order).pack(side="left",
padx=5)

order_list = ttk.Treeview(root, columns=("items", "quantity", "price"), show="headings", height=10)
order_list.heading("items", text="ITEMS")
order_list.heading("quantity", text="QUANTITY")
order_list.heading("price", text="PRICE")
order_list.pack(pady=10)

total_frame = tk.Frame(root)
total_frame.pack(pady=5)

tk.Label(total_frame, text="Total:", font="Arial 12").grid(row=0, column=0, padx=5)
tk.Entry(total_frame, textvariable=total_bill, state="readonly").grid(row=0, column=1, padx=5)

tk.Label(total_frame, text="Payment Type:", font="Arial 12").grid(row=0, column=2, padx=5)
ttk.Combobox(total_frame, textvariable=payment_type, values=["Cash", "UPI", "Card"],
state="readonly").grid(row=0, column=3, padx=5)

tk.Button(root, text="Order Now", bg="orange", command=order_now).pack(pady=20)

tk.Button(root, text="View Sales Graph", bg="orange", command=view_sales_graph).pack(side="left", padx=10)
tk.Button(root, text="View Daily Sales Histogram", bg="yellow", command=view_daily_sales).pack(side="left",
padx=10)

root.mainloop()
```

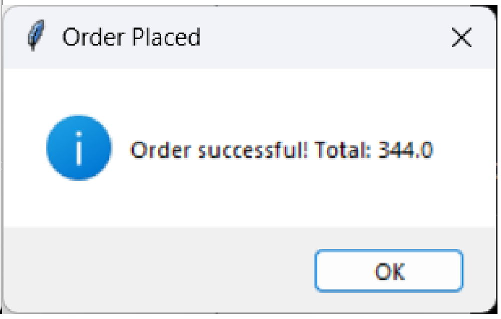


Billing System

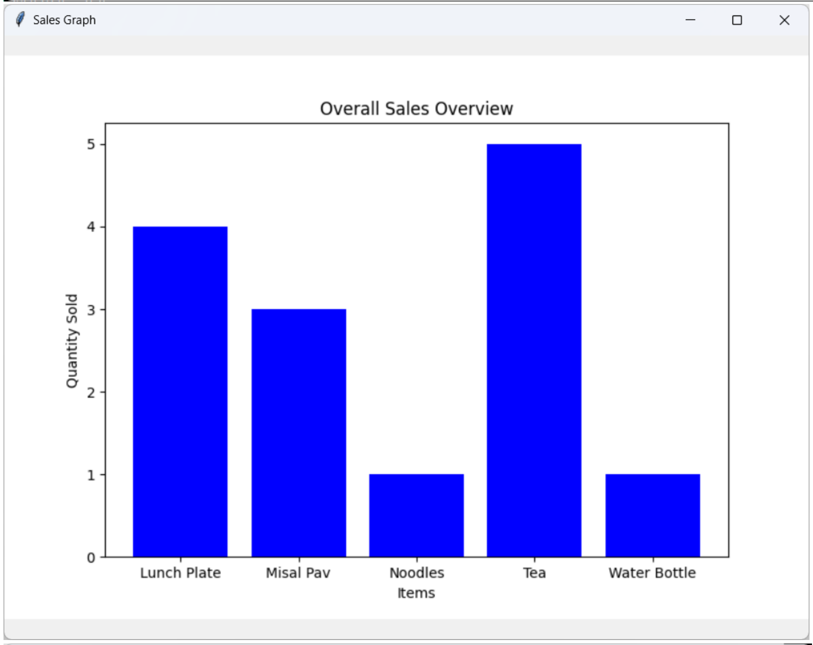
Date:	19-11-2024
Time:	03:55
Bill No:	BN-5971
Menu:	Tea
Quantity:	5
Price:	100

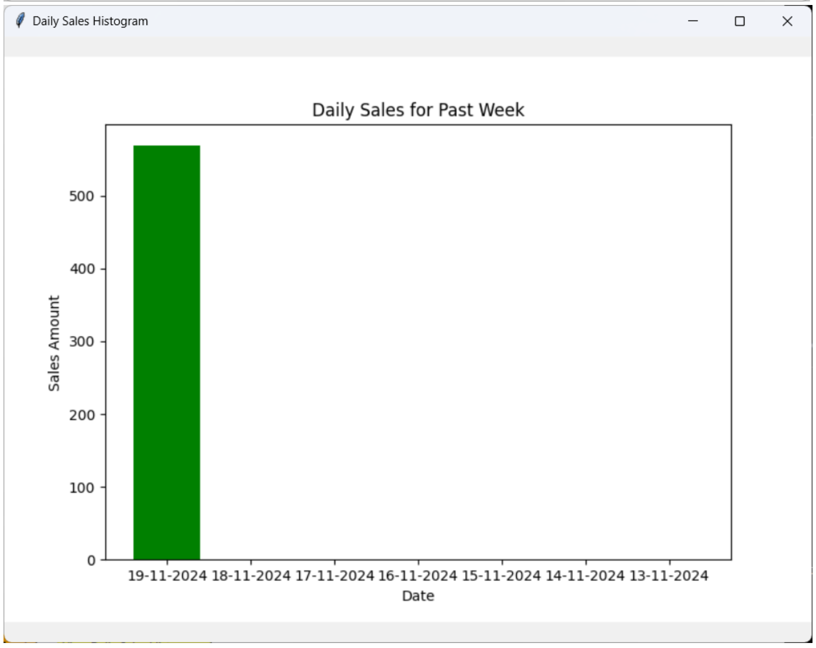
ITEMS	QUANTITY	PRICE
Misal Pav	3	84.0
Lunch Plate	2	160.0
Tea	5	100.0

Total: 344.0 Payment Type: UPI


Order Placed

Order successful! Total: 344.0


Sales Graph
Overall Sales Overview
Quantity Sold
Lunch Plate
Misal Pav
Noodles
Tea
Water Bottle
Items


Daily Sales Histogram
Daily Sales for Past Week
Sales Amount
Date