I didn’t know if you wanted my code in a word file as well but I’m including it just in case.

"""

Author: Jenn Thomason

Date written: 12/12/24

Assignment: Final Project

Short Desc: This application is an Efficient Order Manager designed to help small businesses manage orders efficiently.

"""

import tkinter as tk  # Import the tkinter library for GUI elements

from tkinter import messagebox  # Import messagebox for displaying messages

from PIL import Image, ImageTk  # Import PIL for image handling

import json  # Import json for handling order data

class OrderManagerApp:

    """

    Main application class for the Efficient Order Manager.

    Handles the main window setup and navigation.

    """

    def \_\_init\_\_(self, root):

        """

        Initialize the main application window and set up the UI elements.

        Parameters:

        root (tk.Tk): The main root window object.

        """

        self.root = root  # Main window object

        self.root.title("Efficient Order Manager")  # Set the window title

        self.root.geometry("400x400")  # Set the window size

        # Initialize order data

        self.orders = []  # List to store orders

        # Load images

        self.logo\_photo = self.load\_image("logo.png", (150, 150))  # Company logo image

        self.task\_photo = self.load\_image("task.png", (100, 100))  # Task icon image

        # Menu setup

        self.menu = tk.Menu(root)  # Main menu bar

        root.config(menu=self.menu)  # Configure the root window to use the menu

        # File menu

        self.file\_menu = tk.Menu(self.menu, tearoff=0)  # File menu without tear-off option

        self.menu.add\_cascade(label="File", menu=self.file\_menu)  # Add File menu to the menu bar

        self.file\_menu.add\_command(label="Exit", command=root.quit)  # Add Exit command to the File menu

        # Help menu

        self.help\_menu = tk.Menu(self.menu, tearoff=0)  # Help menu without tear-off option

        self.menu.add\_cascade(label="Help", menu=self.help\_menu)  # Add Help menu to the menu bar

        self.help\_menu.add\_command(label="About", command=self.show\_about)  # Add About command to the Help menu

        # Main window layout

        self.main\_frame = tk.Frame(root)  # Main frame to hold the UI elements

        self.main\_frame.pack(fill=tk.BOTH, expand=True)  # Pack the frame to fill the window

        self.lbl\_logo = tk.Label(self.main\_frame, image=self.logo\_photo, text="Company Logo", compound=tk.BOTTOM)

        self.lbl\_logo.pack(pady=10)  # Display company logo with padding

        self.lbl\_welcome = tk.Label(self.main\_frame, text="Welcome to Efficient Order Manager!", font=("Helvetica", 16))

        self.lbl\_welcome.pack(pady=10)  # Welcome message with custom font

        self.btn\_order\_entry = tk.Button(self.main\_frame, text="Order Entry", command=self.open\_order\_entry)

        self.btn\_order\_entry.pack(pady=5)  # Button to open Order Entry window

        self.btn\_order\_history = tk.Button(self.main\_frame, text="Order History", command=self.open\_order\_history)

        self.btn\_order\_history.pack(pady=5)  # Button to open Order History window

        self.btn\_exit = tk.Button(self.main\_frame, text="Exit", command=self.root.destroy)

        self.btn\_exit.pack(pady=5)  # Exit button to close the application

    def load\_image(self, file\_path, size):

        """

        Load and resize an image.

        Parameters:

        file\_path (str): Path to the image file.

        size (tuple): Desired size of the image.

        Returns:

        ImageTk.PhotoImage: The resized image.

        """

        try:

            image = Image.open(file\_path)  # Open the image file

            image = image.resize(size, Image.Resampling.LANCZOS)  # Resize the image

            return ImageTk.PhotoImage(image)  # Convert to PhotoImage for tkinter

        except IOError:

            messagebox.showerror("Error", f"Unable to load image: {file\_path}")  # Show error message if loading fails

            return ImageTk.PhotoImage(Image.new("RGB", size))  # Return a blank image

    def show\_about(self):

        """

        Display the 'About' information.

        """

        messagebox.showinfo("About", "Efficient Order Manager v1.0\nCreated by Jenn Thomason")  # Show the About message

    def open\_order\_entry(self):

        """

        Open the Order Entry window for entering new orders.

        """

        order\_entry\_window = tk.Toplevel(self.root)  # Create a new window for order entry

        order\_entry\_window.title("Order Entry")  # Set the window title

        order\_entry\_window.geometry("400x400")  # Set the same size as the main window

        # Labels and entry fields for order details

        tk.Label(order\_entry\_window, text="Customer Name").grid(row=0, column=0, padx=10, pady=10)  # Customer name label

        tk.Label(order\_entry\_window, text="Item").grid(row=1, column=0, padx=10, pady=10)  # Item label

        tk.Label(order\_entry\_window, text="Quantity").grid(row=2, column=0, padx=10, pady=10)  # Quantity label

        customer\_name\_entry = tk.Entry(order\_entry\_window)  # Entry for customer name

        item\_entry = tk.Entry(order\_entry\_window)  # Entry for item name

        quantity\_entry = tk.Entry(order\_entry\_window)  # Entry for quantity

        customer\_name\_entry.grid(row=0, column=1, padx=10, pady=10)  # Positioning the entries

        item\_entry.grid(row=1, column=1, padx=10, pady=10)

        quantity\_entry.grid(row=2, column=1, padx=10, pady=10)

        submit\_button = tk.Button(order\_entry\_window, text="Submit", command=lambda: self.submit\_order(customer\_name\_entry, item\_entry, quantity\_entry))

        submit\_button.grid(row=3, column=0, columnspan=2, pady=10)  # Submit button

        exit\_button = tk.Button(order\_entry\_window, text="Exit", command=order\_entry\_window.destroy)

        exit\_button.grid(row=4, column=0, columnspan=2, pady=10)  # Exit button

        self.lbl\_task = tk.Label(order\_entry\_window, image=self.task\_photo, text="Task Icon", compound=tk.BOTTOM)

        self.lbl\_task.grid(row=5, column=0, columnspan=2, pady=10)  # Display task icon

        # Save entries for clearing later

        self.customer\_name\_entry = customer\_name\_entry  # Entry widget for customer name

        self.item\_entry = item\_entry  # Entry widget for item name

        self.quantity\_entry = quantity\_entry  # Entry widget for quantity

    def open\_order\_history(self):

        """

        Open the Order History window to display past orders.

        """

        order\_history\_window = tk.Toplevel(self.root)  # Create a new window for order history

        order\_history\_window.title("Order History")  # Set the window title

        order\_history\_window.geometry("400x400")  # Set the same size as the main window

        tk.Label(order\_history\_window, text="Order History").pack(pady=10)  # Order history header

        # Load orders from file

        self.load\_orders()  # Load orders from a JSON file

        # Display orders

        for order in self.orders:

            order\_text = f"Customer: {order['customer\_name']}, Item: {order['item']}, Quantity: {order['quantity']}"

            tk.Label(order\_history\_window, text=order\_text).pack(pady=5)  # Display each order

        exit\_button = tk.Button(order\_history\_window, text="Exit", command=order\_history\_window.destroy)

        exit\_button.pack(pady=10)  # Exit button

    def load\_orders(self):

        """

        Load orders from a JSON file.

        """

        try:

            with open("orders.json", "r") as file:  # Open the JSON file in read mode

                self.orders = json.load(file)  # Load orders from the file

        except FileNotFoundError:

            self.orders = []  # Initialize as empty if file not found

        except json.JSONDecodeError:

            messagebox.showerror("Error", "Failed to decode the orders file.")  # Show error if decoding fails

    def submit\_order(self, customer\_name\_entry, item\_entry, quantity\_entry):

        """

        Submit a new order and save it to the list of orders.

        """

        customer\_name = customer\_name\_entry.get()  # Get customer name from entry

        item = item\_entry.get()  # Get item name from entry

        quantity = quantity\_entry.get()  # Get quantity from entry

        if not customer\_name or not item or not quantity:  # Check if any field is empty

            messagebox.showwarning("Input Error", "All fields are required!")

            return

        try:

            quantity = int(quantity)  # Ensure quantity is an integer

        except ValueError:

            messagebox.showwarning("Input Error", "Quantity must be a number!")

            return

        # Save the order details

                order = {"customer\_name": customer\_name, "item": item, "quantity": quantity}  # Create a dictionary to store the order details

        self.orders.append(order)  # Add the new order to the list of orders

        # Save orders to a file

        try:

            with open("orders.json", "w") as file:  # Open the JSON file in write mode

                json.dump(self.orders, file)  # Dump the list of orders into the file

            messagebox.showinfo("Success", "Order Submitted Successfully!")  # Show a success message

            # Clear the input fields

            customer\_name\_entry.delete(0, tk.END)  # Clear the customer name entry field

            item\_entry.delete(0, tk.END)  # Clear the item entry field

            quantity\_entry.delete(0, tk.END)  # Clear the quantity entry field

        except IOError:

            messagebox.showerror("Error", "Failed to save the order.")  # Show an error message if saving fails

# Run the application

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

    root = tk.Tk()  # Create the main root window

    app = OrderManagerApp(root)  # Create an instance of the OrderManagerApp

    root.mainloop()  # Run the application