**# importing necessary libraries**

import os

import tkinter as tk

from tkinter import ttk, messagebox, simpledialog

import json

from datetime import datetime

**# defining the main application class**

class TodoList:

    def \_\_init\_\_(self, master):

        # setting up the main window

        self.master = master

        self.master.title("To-Do List")

        self.master.geometry("400x500")

        self.master.configure(bg="#BD5F12")

**# setting up styles**

        style = ttk.Style()

        style.theme\_use('clam')

        style.configure("TFrame", background="#BD5F12")

        style.configure("TButton", padding=10, font=('Times New Roman', 10))

        style.configure("TLabel", background="#BD5F12", font=('Times New Roman', 10))

        style.configure("TEntry", padding=10, font=('Times New Roman', 10))

        style.configure("Treeview", font=('Times New Roman', 10), rowheight=25)

        style.configure("Treeview.Heading", font=('Times New Roman', 10, 'bold'))

        style.map('TButton', background=[('active', '#FFA500')])

**# setting up the main frame**

        self.frame = ttk.Frame(self.master, padding=10, style="TFrame")

        self.frame.pack(fill=tk.BOTH, expand=True)

**# setting up widgets**

        self.task\_var = tk.StringVar()

        self.task\_entry = ttk.Entry(self.frame, textvariable=self.task\_var, width=30, style="TEntry")

        self.task\_entry.grid(row=0, column=0, padx=5, pady=5, sticky="ew")

**# Add Task button**

        self.add\_button = ttk.Button(self.frame, text="Add Task", command=self.add\_task)

        self.add\_button.grid(row=0, column=1, padx=5, pady=10, sticky="ew")

**# Task list display**

        self.task\_tree = ttk.Treeview(self.frame, columns=("Task","Priority","Created At"), show="headings", style="Treeview")

        self.task\_tree.heading("Task", text="Task")

        self.task\_tree.heading("Priority", text="Priority")

        self.task\_tree.heading("Created At", text="Created At")

        self.task\_tree.grid(row=1, column=0, columnspan=2, padx=5, pady=5, sticky="nsew")

**# Color coding based on priority**

        self.task\_tree.tag\_configure("High", background="#FF8783")

        self.task\_tree.tag\_configure("Mid", background="#F7E26D")

        self.task\_tree.tag\_configure("Low", background="#63C763")

**# Adding scrollbar**

        scrollbar = ttk.Scrollbar(self.frame, orient=tk.VERTICAL, command=self.task\_tree.yview)

        scrollbar.grid(row=1, column=2, sticky="ns")

        self.task\_tree.configure(yscrollcommand=scrollbar.set)

**# Control buttons for delete, edit, save, and sort**

        self.delete\_button = ttk.Button(self.frame, text="Delete Task", command=self.delete\_task)

        self.delete\_button.grid(row=2, column=0, padx=5, pady=5, sticky="ew")

        self.edit\_button = ttk.Button(self.frame, text="Edit Task", command=self.edit\_task)

        self.edit\_button.grid(row=2, column=1, padx=5, pady=5, sticky="ew")

        self.save\_button = ttk.Button(self.frame, text="Save Tasks", command=self.save\_tasks)

        self.save\_button.grid(row=3, column=0, columnspan=2, padx=5, pady=5, sticky="ew")

        self.sort\_button = ttk.Button(self.frame, text="Sort by Priority", command=self.sort\_by\_priority)

        self.sort\_button.grid(row=4, column=0, columnspan=2, padx=5, pady=5, sticky="ew")

**# configuring grid weights**

        self.frame.rowconfigure(1, weight=1)

        self.frame.columnconfigure(0, weight=1)

        self.frame.columnconfigure(1, weight=1)

        self.load\_tasks()

**# defining methods for task operations**

    def add\_task(self):

        task = self.task\_var.get().strip()

        if task:

            priority = simpledialog.askstring("Priority", "Enter priority (High,Mid,Low):", parent=self.master)

            if priority and priority.lower() in ["high", "mid", "low"]:

                created\_at = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

                self.task\_tree.insert("", tk.END, values=(task, priority.capitalize(), created\_at))

                self.task\_var.set("")

            else:

                messagebox.showwarning("Invalid Input", "Please enter High, Mid, or Low.")

        else:

            messagebox.showwarning("Warning", "Please enter a task.")

**# method to delete a selected task**

    def delete\_task(self):

        selected\_items = self.task\_tree.selection()

        if selected\_items:

            for item in selected\_items:

                self.task\_tree.delete(item)

        else:

            messagebox.showwarning("Warning", "Please select a task to delete.")

**# method to edit a selected task**

    def edit\_task(self):

        selected\_items = self.task\_tree.selection()

        if selected\_items:

            item = selected\_items[0]

            current\_task, current\_priority, current\_created = self.task\_tree.item(item, "values")

            new\_task = simpledialog.askstring("Edit Task", "Update the task:", initialvalue=current\_task)

            new\_priority = simpledialog.askstring("Edit Priority", "Update priority (High, Mid, Low):", initialvalue=current\_priority)

            if new\_task and new\_priority and new\_priority.lower() in ["high", "mid", "low"]:

                self.task\_tree.item(item, values=(new\_task, current\_created, new\_priority.capitalize()))

            else:

                messagebox.showwarning("Invalid Input", "Please enter valid task and priority.")

        else:

            messagebox.showwarning("Warning", "Please select a task to edit.")

**# method to save tasks to a JSON file**

    def save\_tasks(self):

        tasks = [self.task\_tree.item(child)["values"] for child in self.task\_tree.get\_children()]

        with open("tasks.json", "w") as f:

            json.dump({"tasks": tasks}, f, indent=2)

        messagebox.showinfo("Success", "Tasks saved successfully.")

**# method to load tasks from a JSON file**

    def load\_tasks(self):

        if os.path.exists("tasks.json"):

            try:

                with open("tasks.json", "r") as f:

                    data = json.load(f)

                    tasks = data.get("tasks", [])

                    for task in tasks:

                        if isinstance(task, list) and len(task) == 3:

                            self.task\_tree.insert("", tk.END, values=(task[0], task[1], task[2]), tags=(task[1],))

                        elif isinstance(task, list) and len(task) == 2:

                            self.task\_tree.insert("", tk.END, values=(task[0], task[1], ""), tags=(task[1],))

                        elif isinstance(task, str):  # fallback for old format

                            self.task\_tree.insert("", tk.END, values=(task, "Mid", ""), tags=("Mid",))

            except (json.JSONDecodeError, KeyError):

                messagebox.showerror("Error", "Failed to load tasks. File may be corrupted.")

        else:

            messagebox.showinfo("Info", "No saved tasks found.")

**# method to sort tasks by priority**

    def sort\_by\_priority(self):

        priority\_order = {"High": 1, "Mid": 2, "Low": 3}

        tasks = [

            (priority\_order[item[1]], item[0], item[1], item[2])

            for item in [self.task\_tree.item(i)["values"] for i in self.task\_tree.get\_children()]

            if len(item) == 3

        ]

        tasks.sort()

        for i in self.task\_tree.get\_children():

            self.task\_tree.delete(i)

        for \_, task, priority, created\_at in tasks:

            self.task\_tree.insert("", tk.END, values=(task, priority, created\_at), tags=(priority,))

**# running the application**

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

    root = tk.Tk()

    app = TodoList(root)

    root.mainloop()