import tkinter as tk

from tkinter import messagebox, ttk

from tkcalendar import DateEntry

import json

import os

from datetime import datetime

DATA\_FILE = "gatepass\_data.json"

MENTOR\_USERNAME = "admin"

MENTOR\_PASSWORD = "password"

class GatepassRequest:

def \_init\_(self, name, roll\_number, date, student\_class, reason, status='Pending'):

self.name = name

self.roll\_number = roll\_number

self.date = date

self.student\_class = student\_class

self.reason = reason

self.status = status

def to\_dict(self):

return {

'name': self.name,

'roll\_number': self.roll\_number,

'date': self.date,

'student\_class': self.student\_class,

'reason': self.reason,

'status': self.status

}

@classmethod

def from\_dict(cls, data):

return cls(

name=data['name'],

roll\_number=data['roll\_number'],

date=data['date'],

student\_class=data['student\_class'],

reason=data['reason'],

status=data.get('status', 'Pending')

)

class GatepassManagementSystem:

def \_init\_(self, root):

self.root = root

self.root.title("Gatepass Management System")

self.root.geometry("1400x700")

self.gatepass\_requests = []

self.is\_logged\_in = False

self.load\_data()

self.create\_student\_interface()

self.create\_mentor\_interface()

def load\_data(self):

if os.path.exists(DATA\_FILE):

try:

with open(DATA\_FILE, 'r') as f:

data = json.load(f)

self.gatepass\_requests = [GatepassRequest.from\_dict(item) for item in data]

except Exception as e:

messagebox.showerror("Error", f"Failed to load data: {e}")

def save\_data(self):

try:

with open(DATA\_FILE, 'w') as f:

json.dump([req.to\_dict() for req in self.gatepass\_requests], f, indent=4)

except Exception as e:

messagebox.showerror("Error", f"Failed to save data: {e}")

def create\_student\_interface(self):

student\_frame = ttk.LabelFrame(self.root, text="Student Interface")

student\_frame.grid(row=0, column=0, padx=10, pady=10, sticky="nsew")

ttk.Label(student\_frame, text="Name:").grid(row=0, column=0, padx=5, pady=5, sticky="e")

self.name\_entry = ttk.Entry(student\_frame)

self.name\_entry.grid(row=0, column=1, padx=5, pady=5, sticky="ew")

ttk.Label(student\_frame, text="Roll Number:").grid(row=1, column=0, padx=5, pady=5, sticky="e")

self.roll\_entry = ttk.Entry(student\_frame)

self.roll\_entry.grid(row=1, column=1, padx=5, pady=5, sticky="ew")

ttk.Label(student\_frame, text="Date:").grid(row=2, column=0, padx=5, pady=5, sticky="e")

self.date\_entry = DateEntry(student\_frame, date\_pattern='yyyy-mm-dd', width=18)

self.date\_entry.grid(row=2, column=1, padx=5, pady=5, sticky="w")

ttk.Label(student\_frame, text="Class:").grid(row=3, column=0, padx=5, pady=5, sticky="e")

self.class\_entry = ttk.Combobox(student\_frame, values=["CSE-DS-A", "CSE-DS-B"], state="readonly")

self.class\_entry.grid(row=3, column=1, padx=5, pady=5, sticky="ew")

ttk.Label(student\_frame, text="Reason:").grid(row=4, column=0, padx=5, pady=5, sticky="ne")

self.reason\_text = tk.Text(student\_frame, height=4, width=30)

self.reason\_text.grid(row=4, column=1, padx=5, pady=5, sticky="ew")

submit\_btn = ttk.Button(student\_frame, text="Submit", command=self.submit\_request)

submit\_btn.grid(row=5, column=0, columnspan=2, pady=10)

def submit\_request(self):

name = self.name\_entry.get().strip()

roll\_number = self.roll\_entry.get().strip()

date = self.date\_entry.get\_date().strftime('%Y-%m-%d')

student\_class = self.class\_entry.get().strip()

reason = self.reason\_text.get("1.0", tk.END).strip()

if not all([name, roll\_number, date, student\_class, reason]):

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

return

new\_request = GatepassRequest(name, roll\_number, date, student\_class, reason)

self.gatepass\_requests.append(new\_request)

self.save\_data()

if self.is\_logged\_in:

self.update\_mentor\_treeview()

def create\_mentor\_interface(self):

mentor\_frame = ttk.LabelFrame(self.root, text="Mentor Interface")

mentor\_frame.grid(row=0, column=1, padx=10, pady=10, sticky="nsew")

ttk.Label(mentor\_frame, text="Username:").grid(row=0, column=0, padx=5, pady=5, sticky="e")

self.username\_entry = ttk.Entry(mentor\_frame)

self.username\_entry.grid(row=0, column=1, padx=5, pady=5, sticky="ew")

ttk.Label(mentor\_frame, text="Password:").grid(row=1, column=0, padx=5, pady=5, sticky="e")

self.password\_entry = ttk.Entry(mentor\_frame, show="\*")

self.password\_entry.grid(row=1, column=1, padx=5, pady=5, sticky="ew")

login\_btn = ttk.Button(mentor\_frame, text="Login", command=self.mentor\_login)

login\_btn.grid(row=2, column=0, columnspan=2, pady=10)

self.mentor\_treeview = ttk.Treeview(mentor\_frame, columns=("Name", "Roll Number", "Date", "Class", "Reason", "Status"), show="headings")

for col in self.mentor\_treeview["columns"]:

self.mentor\_treeview.heading(col, text=col)

self.mentor\_treeview.column(col, width=100, anchor="center")

self.mentor\_treeview.grid(row=3, column=0, columnspan=2, padx=5, pady=5, sticky="nsew")

accept\_btn = ttk.Button(mentor\_frame, text="Accept", command=lambda: self.change\_request\_status("Accepted"))

accept\_btn.grid(row=4, column=0, pady=10)

reject\_btn = ttk.Button(mentor\_frame, text="Reject", command=lambda: self.change\_request\_status("Rejected"))

reject\_btn.grid(row=4, column=1, pady=10)

def mentor\_login(self):

username = self.username\_entry.get().strip()

password = self.password\_entry.get().strip()

if username == MENTOR\_USERNAME and password == MENTOR\_PASSWORD:

self.is\_logged\_in = True

messagebox.showinfo("Login Success", "Mentor logged in successfully.")

self.update\_mentor\_treeview()

else:

messagebox.showerror("Login Failed", "Invalid credentials.")

def update\_mentor\_treeview(self):

self.mentor\_treeview.delete(\*self.mentor\_treeview.get\_children())

for req in self.gatepass\_requests:

self.mentor\_treeview.insert("", "end", values=(req.name, req.roll\_number, req.date, req.student\_class, req.reason, req.status))

def change\_request\_status(self, status):

selected\_item = self.mentor\_treeview.selection()

if selected\_item:

index = self.mentor\_treeview.index(selected\_item[0])

self.gatepass\_requests[index].status = status

self.save\_data()

self.update\_mentor\_treeview()

def main():

root = tk.Tk()

app = GatepassManagementSystem(root)

root.mainloop()

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

    main()