import tkinter as tk

from tkinter import font, messagebox, ttk

from PIL import Image, ImageTk

import os

import sqlite3

import tkinter.font as tkFont

import datetime

class PadreGarciaApp:

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

self.root = root

self.root.title("Padre Garcia Polytechnic College")

self.root.configure(bg="#ffffff")

screen\_width = self.root.winfo\_screenwidth()

screen\_height = self.root.winfo\_screenheight()

width = int(screen\_width \* 0.55)

height = int(screen\_height \* 0.70)

min\_width, min\_height = 600, 600

width = max(width, min\_width)

height = max(height, min\_height)

self.root.geometry(f"{width}x{height}")

self.root.minsize(min\_width, min\_height)

self.root.grid\_rowconfigure(0, weight=1)

self.root.grid\_columnconfigure(0, weight=1) # Add this line

self.main\_frame = tk.Frame(self.root, bg="#ffffff", bd=1, relief="solid")

self.main\_frame.grid(row=0, column=0, sticky="nsew", padx=10, pady=10) # Reduce padding

self.main\_frame.grid\_rowconfigure(0, weight=1) # Add these two lines

self.main\_frame.grid\_columnconfigure(0, weight=1)

# Connect to DB

self.conn = sqlite3.connect('padre\_garcia\_polytechnic.db')

self.cursor = self.conn.cursor()

# Uncomment the next line to reset database schema; WARNING: This deletes all data

# self.reset\_database()

self.create\_database()

self.students = []

self.editing\_student\_no = None

self.payment\_terms = [

"Initial", "Prelim", "Midterm", "Final",

"Payment 5", "Payment 6", "Payment 7",

"Payment 8", "Payment 9", "Payment 10"

]

self.tuition\_fees = {

"Garciano": {

"First Semester": {

"Bachelor of Science in Computer Science": 4500,

"Bachelor of Science in Management Accounting": 4500,

"Bachelor of Science in Public Administration": 4500,

"Bachelor of Science in Criminology": 4500

},

"Second Semester": {

"Bachelor of Science in Computer Science": 4500,

"Bachelor of Science in Management Accounting": 4500,

"Bachelor of Science in Public Administration": 4500,

"Bachelor of Science in Criminology": 4500

}

},

"Non-Garciano": {

"First Semester": {

"Bachelor of Science in Computer Science": 13400,

"Bachelor of Science in Management Accounting": 15700,

"Bachelor of Science in Public Administration": 15500,

"Bachelor of Science in Criminology": 14700

},

"Second Semester": {

"Bachelor of Science in Computer Science": 13700,

"Bachelor of Science in Management Accounting": 14900,

"Bachelor of Science in Public Administration": 14900,

"Bachelor of Science in Criminology": 14700

}

}

}

self.load\_students\_from\_db()

self.load\_welcome\_view()

def reset\_database(self):

# WARNING: This will erase all data and recreate tables

self.cursor.execute('DROP TABLE IF EXISTS payments')

self.cursor.execute('DROP TABLE IF EXISTS students')

self.conn.commit()

def create\_database(self):

# Create tables if not exist with all needed columns

self.cursor.execute('''

CREATE TABLE IF NOT EXISTS students (

id INTEGER PRIMARY KEY AUTOINCREMENT,

last\_name TEXT NOT NULL,

first\_name TEXT NOT NULL,

middle\_name TEXT,

student\_no TEXT UNIQUE NOT NULL,

password TEXT,

contact\_number TEXT,

gmail TEXT,

course TEXT,

section TEXT,

year TEXT,

semester TEXT,

category TEXT,

tuition\_fee REAL

)

''')

self.cursor.execute('''

CREATE TABLE IF NOT EXISTS payments (

id INTEGER PRIMARY KEY AUTOINCREMENT,

student\_no TEXT NOT NULL,

term TEXT NOT NULL,

amount REAL NOT NULL,

date TEXT,

FOREIGN KEY (student\_no) REFERENCES students (student\_no)

)

''')

self.conn.commit()

self.cursor.execute('''

CREATE TABLE IF NOT EXISTS enrollments (

id INTEGER PRIMARY KEY AUTOINCREMENT,

student\_no TEXT NOT NULL,

year TEXT,

semester TEXT,

date\_enrolled TEXT,

FOREIGN KEY (student\_no) REFERENCES students (student\_no)

)

''')

self.conn.commit()

def load\_students\_from\_db(self):

try:

self.cursor.execute("SELECT id, last\_name, first\_name, middle\_name, student\_no, password, contact\_number, gmail, course, section, year, semester, category, tuition\_fee FROM students")

except sqlite3.OperationalError as e:

messagebox.showerror("Database Error", f"Database schema mismatch detected.\n{e}\n\n"

"To fix this, please delete the 'padre\_garcia\_polytechnic.db' file "

"in your project directory so the app can recreate it with the correct schema.")

self.root.destroy()

return

rows = self.cursor.fetchall()

self.students = []

for row in rows:

student\_data = {

"id": row[0],

"last\_name": row[1],

"first\_name": row[2],

"middle\_name": row[3],

"student\_no": row[4],

"password": row[5] or "",

"contact\_number": row[6] or "",

"gmail": row[7] or "",

"course": row[8],

"section": row[9],

"year": row[10],

"semester": row[11],

"category": row[12],

"tuition\_fee": row[13] or 0

}

self.students.append(student\_data)

def populate\_enrollments\_for\_existing\_students(self):

self.cursor.execute("SELECT student\_no, year, semester FROM students")

students = self.cursor.fetchall()

for student\_no, year, semester in students:

# Check if enrollment already exists for this year/semester

self.cursor.execute(

"SELECT 1 FROM enrollments WHERE student\_no = ? AND year = ? AND semester = ?",

(student\_no, year, semester)

)

if not self.cursor.fetchone():

self.cursor.execute(

"INSERT INTO enrollments (student\_no, year, semester, date\_enrolled) VALUES (?, ?, ?, ?)",

(student\_no, year, semester, datetime.date.today().isoformat())

)

self.conn.commit()

def load\_welcome\_view(self):

self.current\_view = "welcome"

for widget in self.main\_frame.winfo\_children():

widget.destroy()

# --- Top navigation bar ---

nav = tk.Frame(self.main\_frame, bg="#f8fafc", height=56)

nav.pack(fill='x', side='top')

lbl\_brand = tk.Label(nav, text="Padre Garcia Polytechnic College", font=("Segoe UI", 16, "bold"), bg="#f8fafc", fg="#22223b")

lbl\_brand.pack(side='left', padx=24, pady=12)

menu\_btn = tk.Button(

nav,

text="≡",

font=("Segoe UI", 24, "bold"),

bg="#f8fafc",

fg="#22223b",

bd=0,

activebackground="#e5e7eb",

activeforeground="#22223b",

cursor="hand2",

command=self.show\_welcome\_menu

)

menu\_btn.pack(side='right', padx=24, pady=8)

menu\_btn.tooltip = tk.Label(nav, text="Menu", bg="#f8fafc", fg="#22223b", font=("Segoe UI", 9))

def on\_enter(e): menu\_btn.tooltip.place(x=nav.winfo\_width()-60, y=10)

def on\_leave(e): menu\_btn.tooltip.place\_forget()

menu\_btn.bind("<Enter>", on\_enter)

menu\_btn.bind("<Leave>", on\_leave)

# --- Main content area ---

bg\_frame = tk.Frame(self.main\_frame, bg="#e5e7eb")

bg\_frame.pack(expand=True, fill='both')

card = tk.Frame(bg\_frame, bg="#fff", bd=2, relief="groove")

card.place(relx=0.5, rely=0.5, anchor="center", width=500, height=420)

logo\_size = 120

logo\_path = r"C:\Users\Angel\_P\_Reyes\OneDrive\Desktop\tk\_project\Images\Images\logo\_real.png"

if os.path.exists(logo\_path):

img = Image.open(logo\_path)

img = img.resize((logo\_size, logo\_size))

self.logo\_img = ImageTk.PhotoImage(img)

lbl\_logo = tk.Label(card, image=self.logo\_img, bg="#fff")

lbl\_logo.pack(pady=(30, 10))

else:

logo\_canvas = tk.Canvas(card, width=logo\_size, height=logo\_size, bg="#fff", highlightthickness=0)

logo\_canvas.create\_text(logo\_size // 2, logo\_size // 2, text="Logo", fill="#374151", font=("Segoe UI", 24, "bold"))

logo\_canvas.pack(pady=(30, 10))

welcome\_font = font.Font(family="Segoe UI", size=22, weight="bold")

lbl\_welcome = tk.Label(card,

text="Welcome to Padre Garcia Polytechnic College",

font=welcome\_font, bg="#fff", fg="#1f2937", justify="center", wraplength=400)

lbl\_welcome.pack(pady=(0, 8), padx=20)

tagline\_font = font.Font(family="Segoe UI", size=12, slant="italic")

lbl\_tagline = tk.Label(card,

text="Taga-PGPC Ako: Matalino, Disiplinado, \n Mabuting Tao, Ipinagmamalaki ko!",

font=tagline\_font, bg="#fff", fg="#6b7280")

lbl\_tagline.pack(pady=(0, 20))

# Social/contact info

lbl\_contact = tk.Label(card, text="Contact: padregarciapolytechniccollege@gmail.com", font=("Segoe UI", 10), bg="#fff", fg="#9ca3af")

lbl\_contact.pack(side="bottom", pady=(0, 16))

# Footer

lbl\_footer = tk.Label(bg\_frame, text="© 2025 Padre Garcia Polytechnic College", font=("Segoe UI", 9), bg="#e5e7eb", fg="#9ca3af")

lbl\_footer.pack(side="bottom", pady=(0, 8))

# ...existing code...

def show\_welcome\_menu(self):

# Prevent multiple menus

if hasattr(self, 'dropdown\_menu') and self.dropdown\_menu.winfo\_exists():

return

# Get the position of the hamburger menu button

menu\_btn = None

for widget in self.main\_frame.winfo\_children():

if isinstance(widget, tk.Frame):

for child in widget.winfo\_children():

if isinstance(child, tk.Button) and child['text'] == "≡":

menu\_btn = child

break

if not menu\_btn:

return

# Calculate position

x = menu\_btn.winfo\_rootx()

y = menu\_btn.winfo\_rooty() + menu\_btn.winfo\_height()

dropdown\_width = 320

dropdown\_height = 400

screen\_width = self.root.winfo\_screenwidth()

if x + dropdown\_width > screen\_width:

x = screen\_width - dropdown\_width - 10

self.dropdown\_menu = tk.Toplevel(self.root)

self.dropdown\_menu.overrideredirect(True)

self.dropdown\_menu.geometry(f"{dropdown\_width}x{dropdown\_height}+{x}+{y}")

self.dropdown\_menu.configure(bg="#f3f4f6", bd=2, relief="raised")

def close\_dropdown():

if self.dropdown\_menu.winfo\_exists():

self.dropdown\_menu.destroy()

# ...inside show\_welcome\_menu...

close\_btn = tk.Button(

self.dropdown\_menu,

text="✕",

font=("Segoe UI", 10, "bold"),

bg="#f3f4f6",

fg="#22223b",

bd=0,

relief="flat",

activebackground="#e5e7eb",

activeforeground="#dc2626",

cursor="hand2",

command=self.dropdown\_menu.destroy,

padx=0,

pady=0,

width=2, # Just enough for the X

height=1 # Just enough for the X

)

close\_btn.place(relx=1.0, rely=0.0, anchor="ne", x=-4, y=4)

# --- Admin Login Button ---

def toggle\_admin\_login\_panel():

# Hide if already shown

if hasattr(self, 'admin\_login\_panel') and self.admin\_login\_panel.winfo\_exists():

self.admin\_login\_panel.destroy()

return

# Hide student panel if open

if hasattr(self, 'student\_login\_panel') and self.student\_login\_panel.winfo\_exists():

self.student\_login\_panel.destroy()

# Show admin login panel

self.admin\_login\_panel = tk.Frame(self.dropdown\_menu, bg="#f3f4f6")

self.admin\_login\_panel.pack(fill='x', padx=40, pady=(0, 10))

tk.Label(self.admin\_login\_panel, text="Admin Login", font=("Segoe UI", 11, "bold"), bg="#f3f4f6").pack(anchor="w")

# Label above username entry

tk.Label(self.admin\_login\_panel, text="Admin Username", font=("Segoe UI", 10), bg="#f3f4f6", anchor='w').pack(fill='x', pady=(0, 2))

entry\_username = ttk.Entry(self.admin\_login\_panel, font=("Segoe UI", 11))

entry\_username.pack(fill='x', pady=(0, 6))

# ...inside toggle\_admin\_login\_panel in show\_welcome\_menu...

# Label above password entry

tk.Label(self.admin\_login\_panel, text="Password", font=("Segoe UI", 10), bg="#f3f4f6", anchor='w').pack(fill='x', pady=(0, 2))

pw\_frame = tk.Frame(self.admin\_login\_panel, bg="#f3f4f6")

pw\_frame.pack(fill='x', pady=(0, 8))

entry\_password = ttk.Entry(pw\_frame, font=("Segoe UI", 11), show="\*")

entry\_password.pack(side='left', fill='x', expand=True)

show\_pw = tk.BooleanVar(value=False)

def toggle\_password():

if show\_pw.get():

entry\_password.config(show="\*")

btn\_eye.config(text="👁️") # Eye with slash (or use a suitable Unicode)

show\_pw.set(False)

else:

entry\_password.config(show="")

btn\_eye.config(text="👁️") # Eye open

show\_pw.set(True)

btn\_eye = tk.Button(

pw\_frame,

text="👁️‍🗨️", # Start with eye-slash

font=("Segoe UI", 10),

bg="#f3f4f6",

bd=0,

relief="flat",

command=toggle\_password,

cursor="hand2"

)

btn\_eye.pack(side='right')

def do\_admin\_login():

username = entry\_username.get()

password = entry\_password.get()

if username == "admin" and password == "password123":

close\_dropdown()

messagebox.showinfo("Login Successful", "Welcome, Admin!")

self.load\_manage\_student\_panel()

else:

messagebox.showerror("Login Failed", "Invalid Admin Username or Password.")

ttk.Button(self.admin\_login\_panel, text="Login", command=do\_admin\_login, style="Primary.TButton").pack(fill='x', pady=(0, 4))

btn\_admin = ttk.Button(

self.dropdown\_menu,

text="Admin Login ▼",

command=toggle\_admin\_login\_panel,

style="Primary.TButton"

)

btn\_admin.pack(fill='x', padx=20, pady=(18, 4), ipady=6)

# --- Student Login Button ---

def toggle\_student\_login\_panel():

# Hide if already shown

if hasattr(self, 'student\_login\_panel') and self.student\_login\_panel.winfo\_exists():

self.student\_login\_panel.destroy()

return

# Hide admin panel if open

if hasattr(self, 'admin\_login\_panel') and self.admin\_login\_panel.winfo\_exists():

self.admin\_login\_panel.destroy()

# Show student login panel

self.student\_login\_panel = tk.Frame(self.dropdown\_menu, bg="#f3f4f6")

self.student\_login\_panel.pack(fill='x', padx=40, pady=(0, 10))

tk.Label(self.student\_login\_panel, text="Student Login", font=("Segoe UI", 11, "bold"), bg="#f3f4f6").pack(anchor="w")

# Label above student number entry

tk.Label(self.student\_login\_panel, text="Student Number", font=("Segoe UI", 10), bg="#f3f4f6", anchor='w').pack(fill='x', pady=(0, 2))

entry\_student\_no = ttk.Entry(self.student\_login\_panel, font=("Segoe UI", 11))

entry\_student\_no.pack(fill='x', pady=(0, 6))

# Label above password entry

tk.Label(self.student\_login\_panel, text="Password", font=("Segoe UI", 10), bg="#f3f4f6", anchor='w').pack(fill='x', pady=(0, 2))

entry\_password = ttk.Entry(self.student\_login\_panel, font=("Segoe UI", 11), show="\*")

entry\_password.pack(fill='x', pady=(0, 8))

def do\_student\_login():

student\_no = entry\_student\_no.get()

password = entry\_password.get()

self.cursor.execute('SELECT password, first\_name FROM students WHERE student\_no = ?', (student\_no,))

row = self.cursor.fetchone()

if row and row[0] == password:

close\_dropdown()

messagebox.showinfo("Login Successful", f"Welcome, {row[1]}!")

self.show\_student\_dashboard(student\_no)

else:

messagebox.showerror("Login Failed", "Invalid student number or password.")

ttk.Button(self.student\_login\_panel, text="Login", command=do\_student\_login, style="Primary.TButton").pack(fill='x', pady=(0, 4))

btn\_student = ttk.Button(

self.dropdown\_menu,

text="Student Login ▼",

command=toggle\_student\_login\_panel,

style="Primary.TButton"

)

btn\_student.pack(fill='x', padx=20, pady=4, ipady=6)

# --- About Button ---

btn\_about = ttk.Button(

self.dropdown\_menu,

text="About",

command=lambda: [close\_dropdown(), self.show\_about\_dialog()],

style="Primary.TButton"

)

btn\_about.pack(fill='x', padx=20, pady=4, ipady=6)

def show\_about\_dialog(self):

messagebox.showinfo("About", "Padre Garcia Polytechnic College\nStudent Management System\n© 2025")

def load\_login\_view(self):

self.current\_view = "login"

for widget in self.main\_frame.winfo\_children():

widget.destroy()

nav = tk.Frame(self.main\_frame, bg="#f8fafc", height=40)

nav.pack(fill='x', side='top')

lbl\_brand = tk.Label(nav, text="Padre Garcia Polytechnic College", font=("Segoe UI", 13, "bold"), bg="#f8fafc", fg="#22223b")

lbl\_brand.pack(side='left', padx=12, pady=8)

# Centered card frame

center = tk.Frame(self.main\_frame, bg="#ffffff")

center.pack(expand=True, fill='both')

# Simulated shadow (make it wider and taller)

shadow = tk.Frame(center, bg="#e5e7eb")

shadow.place(relx=0.5, rely=0.5, anchor="center", width=620, height=260)

# Bigger card for logo + form

card = tk.Frame(center, bg="#f3f4f6", bd=2, relief="groove")

card.place(relx=0.5, rely=0.5, anchor="center", width=610, height=250)

# --- Side-by-side layout ---

card.grid\_rowconfigure(0, weight=1)

card.grid\_columnconfigure(1, weight=1)

# Logo section (left)

logo\_path = "path\_to\_small\_logo.png" # Change to your logo path

circle\_diameter = 130

if os.path.exists(logo\_path):

logo\_img = Image.open(logo\_path).resize((100, 100))

self.login\_logo\_img = ImageTk.PhotoImage(logo\_img)

logo\_canvas = tk.Canvas(card, width=circle\_diameter, height=circle\_diameter, bg="#f3f4f6", highlightthickness=0)

# Draw circle

logo\_canvas.create\_oval(5, 5, circle\_diameter-5, circle\_diameter-5, fill="#e5e7eb", outline="#e5e7eb")

# Place logo image in the center

logo\_canvas.create\_image(circle\_diameter//2, circle\_diameter//2, image=self.login\_logo\_img)

logo\_canvas.grid(row=0, column=0, rowspan=4, padx=(36, 18), pady=36, sticky="n")

else:

logo\_canvas = tk.Canvas(card, width=circle\_diameter, height=circle\_diameter, bg="#f3f4f6", highlightthickness=0)

logo\_canvas.create\_oval(5, 5, circle\_diameter-5, circle\_diameter-5, fill="#e5e7eb", outline="#e5e7eb")

logo\_canvas.create\_text(circle\_diameter//2, circle\_diameter//2, text="Logo", fill="#4a4e69", font=("Segoe UI", 18, "bold"))

logo\_canvas.grid(row=0, column=0, rowspan=4, padx=(36, 18), pady=36, sticky="n")

# Login form section (right)

lbl\_login = tk.Label(card, text="Log In", font=("Segoe UI", 20, "bold"), bg="#f3f4f6", fg="#22223b")

lbl\_login.grid(row=0, column=1, sticky="w", pady=(36, 4), padx=(0, 36))

lbl\_sub = tk.Label(card, text="Access your account", font=("Segoe UI", 12), bg="#f3f4f6", fg="#6b7280")

lbl\_sub.grid(row=1, column=1, sticky="w", padx=(0, 36), pady=(0, 16))

btn\_style = {"style": "Primary.TButton", "width": 20}

#btn\_admin = ttk.Button(card, text="Admin Login", command=self.load\_admin\_login\_panel, \*\*btn\_style)

#btn\_admin.grid(row=2, column=1, sticky="ew", padx=(0, 36), pady=(0, 8), ipady=8)

btn\_student = ttk.Button(card, text="Student Login", command=self.load\_student\_login\_panel, \*\*btn\_style)

btn\_student.grid(row=3, column=1, sticky="ew", padx=(0, 36), pady=(0, 16), ipady=8)

lbl\_footer = tk.Label(center, text="© 2025 Padre Garcia Polytechnic College", font=("Segoe UI", 9), bg="#ffffff", fg="#9ca3af")

lbl\_footer.pack(side="bottom", pady=(10, 0))

self.apply\_styles()

def apply\_styles(self):

style = ttk.Style()

style.theme\_use('clam')

# ...existing styles...

style.configure('Primary.TButton',

font=("Segoe UI", 12, "bold"),

padding=(8, 4),

background="#22223b",

foreground="#fff",

borderwidth=0,

relief="flat"

)

style.map('Primary.TButton',

background=[('active', '#4a4e69')],

foreground=[('active', '#fff')]

)

def load\_student\_login\_panel(self):

self.current\_view = "student\_login"

for widget in self.main\_frame.winfo\_children():

widget.destroy()

nav = tk.Frame(self.main\_frame, bg="#f3f4f6", height=48)

nav.pack(fill='x', side='top')

# Back arrow button (upper left)

btn\_back = tk.Button(

nav,

text="←", # Unicode left arrow

font=("Segoe UI", 18, "bold"),

bg="#f3f4f6",

fg="#22223b",

bd=0,

activebackground="#e5e7eb",

activeforeground="#22223b",

cursor="hand2",

command=self.load\_welcome\_view

)

btn\_back.pack(side='left', padx=(12, 0), pady=8)

lbl\_brand = tk.Label(nav, text="Padre Garcia Polytechnic College", font=("Segoe UI", 16, "bold"), bg="#f3f4f6", fg="#1f2937")

lbl\_brand.pack(side='left', padx=12, pady=12)

center = tk.Frame(self.main\_frame, bg="#ffffff")

center.pack(expand=True, fill='both')

lbl\_title = tk.Label(center, text="Student Login", font=("Segoe UI", 18, "bold"), bg="#fff", fg="#1f2937")

lbl\_title.pack(pady=(30, 20))

entry\_font = font.Font(family="Segoe UI", size=12)

frm\_inputs = tk.Frame(center, bg="#fff")

frm\_inputs.pack(pady=(0, 20), padx=60, fill="x")

lbl\_student\_no = tk.Label(frm\_inputs, text="Student Number:", font=entry\_font, bg="#fff", anchor='w')

lbl\_student\_no.pack(fill='x', pady=(0, 5))

self.entry\_student\_no\_login = ttk.Entry(frm\_inputs, font=entry\_font)

self.entry\_student\_no\_login.pack(fill='x', pady=(0, 10))

lbl\_password = tk.Label(frm\_inputs, text="Password:", font=entry\_font, bg="#fff", anchor='w')

lbl\_password.pack(fill='x', pady=(0, 5))

self.entry\_password\_login = ttk.Entry(frm\_inputs, font=entry\_font, show='\*')

self.entry\_password\_login.pack(fill='x', pady=(0, 20))

btn\_login = ttk.Button(frm\_inputs, text="Login", command=self.validate\_student\_login, style="Primary.TButton")

btn\_login.pack(fill='x', pady=(0, 10))

#btn\_back = ttk.Button(frm\_inputs, text="Back", command=self.load\_welcome\_view, style="Danger.TButton")

# --- Add this block below the login button ---

frm\_signup = tk.Frame(center, bg="#fff")

frm\_signup.pack(pady=(0, 20))

lbl\_no\_account = tk.Label(frm\_signup, text="Don't have an account?", font=("Segoe UI", 10), bg="#fff", fg="#6b7280")

lbl\_no\_account.pack(side="left")

btn\_signup = tk.Button(frm\_signup, text="Sign up", font=("Segoe UI", 10, "underline"), fg="#2563eb", bg="#fff", bd=0, cursor="hand2",

command=lambda: self.load\_student\_registration\_panel(from\_admin=False))

btn\_signup.pack(side="left", padx=(5, 0))

def validate\_student\_login(self):

student\_no = self.entry\_student\_no\_login.get().strip()

password = self.entry\_password\_login.get().strip()

if not student\_no or not password:

messagebox.showerror("Login Failed", "Please enter both student number and password.")

return

self.cursor.execute('SELECT password, first\_name FROM students WHERE student\_no = ?', (student\_no,))

row = self.cursor.fetchone()

if row and row[0] == password:

messagebox.showinfo("Login Successful", f"Welcome, {row[1]}!")

self.show\_student\_dashboard(student\_no)

else:

messagebox.showerror("Login Failed", "Invalid student number or password.")

def show\_student\_dashboard(self, student\_no):

self.current\_view = "student\_dashboard"

for widget in self.main\_frame.winfo\_children():

widget.destroy()

# Sidebar frame (left)

sidebar = tk.Frame(self.main\_frame, bg="#f3f4f6", width=220)

sidebar.pack(side="left", fill="y")

btn\_profile = ttk.Button(sidebar, text="View Profile", style="Primary.TButton",

command=lambda: self.show\_student\_profile\_content(student\_no))

btn\_profile.pack(fill='x', padx=30, pady=(40, 10), ipady=10)

btn\_payment = ttk.Button(sidebar, text="View Payment Record", style="Success.TButton",

command=lambda: self.show\_student\_payment\_content(student\_no))

btn\_payment.pack(fill='x', padx=30, pady=10, ipady=10)

btn\_logout = ttk.Button(sidebar, text="Logout", style="Danger.TButton",

command=self.load\_welcome\_view)

btn\_logout.pack(fill='x', padx=30, pady=(10, 20), ipady=10)

# Main content frame (right)

self.student\_right\_frame = tk.Frame(self.main\_frame, bg="#ffffff")

self.student\_right\_frame.pack(side="left", expand=True, fill="both")

# Show profile by default

self.show\_student\_profile\_content(student\_no)

def show\_student\_profile\_content(self, student\_no):

for widget in self.student\_right\_frame.winfo\_children():

widget.destroy()

student = next((s for s in self.students if s.get("student\_no") == student\_no), None)

if not student:

messagebox.showerror("Profile", "Student not found.")

self.load\_login\_view()

return

lbl\_title = tk.Label(self.student\_right\_frame, text="Student Profile", font=("Segoe UI", 20, "bold"), bg="#ffffff", fg="#1f2937")

lbl\_title.pack(pady=(40, 20))

info\_font = font.Font(family="Segoe UI", size=13)

profile\_text = (

f"Name: {student.get('first\_name','')} {student.get('middle\_name','')} {student.get('last\_name','')}\n"

f"Student No: {student.get('student\_no','')}\n"

f"Contact: {student.get('contact\_number','')}\n"

f"Gmail: {student.get('gmail','')}\n"

f"Course: {student.get('course','')}\n"

f"Year: {student.get('year','')}\n"

f"Section: {student.get('section','')}\n"

f"Semester: {student.get('semester','')}\n"

f"Category: {student.get('category','')}\n"

f"Tuition Fee: ₱{student.get('tuition\_fee',0):,.2f}"

)

lbl\_profile = tk.Label(self.student\_right\_frame, text=profile\_text, font=info\_font, bg="#ffffff", fg="#22223b", justify="left", anchor="w")

lbl\_profile.pack(pady=(0, 20), padx=60, anchor="w")

def show\_student\_payment\_content(self, student\_no):

for widget in self.student\_right\_frame.winfo\_children():

widget.destroy()

student = next((s for s in self.students if s.get("student\_no") == student\_no), None)

if not student:

messagebox.showerror("Payment Record", "Student not found.")

self.load\_login\_view()

return

lbl\_title = tk.Label(self.student\_right\_frame, text="Payment Record", font=("Segoe UI", 20, "bold"), bg="#ffffff", fg="#1f2937")

lbl\_title.pack(pady=(40, 20))

# Dropdowns for year and semester

year\_options = ["1st Year", "2nd Year", "3rd Year", "4th Year"]

semester\_options = ["First Semester", "Second Semester"]

year\_var = tk.StringVar(value=student.get("year", year\_options[0]))

semester\_var = tk.StringVar(value=student.get("semester", semester\_options[0]))

select\_frame = tk.Frame(self.student\_right\_frame, bg="#ffffff")

select\_frame.pack(pady=(0, 10), padx=60, anchor='w')

tk.Label(select\_frame, text="Year:", bg="#ffffff").pack(side="left")

year\_dropdown = ttk.Combobox(select\_frame, textvariable=year\_var, values=year\_options, state="readonly", width=12)

year\_dropdown.pack(side="left", padx=(5, 20))

tk.Label(select\_frame, text="Semester:", bg="#ffffff").pack(side="left")

semester\_dropdown = ttk.Combobox(select\_frame, textvariable=semester\_var, values=semester\_options, state="readonly", width=16)

semester\_dropdown.pack(side="left", padx=(5, 0))

columns = ("Term", "Amount (₱)", "Date")

payment\_tree = ttk.Treeview(self.student\_right\_frame, columns=columns, show="headings", height=12)

payment\_tree.pack(fill='both', expand=True, padx=60, pady=10)

for col in columns:

payment\_tree.heading(col, text=col)

payment\_tree.column(col, anchor='center')

def load\_payments\_for\_period():

# Clear previous rows

for row in payment\_tree.get\_children():

payment\_tree.delete(row)

selected\_year = year\_var.get()

selected\_semester = semester\_var.get()

payment\_data = {term: {"Amount": 0.0, "Date": ""} for term in self.payment\_terms}

self.cursor.execute(

'SELECT term, amount, date FROM payments WHERE student\_no = ? AND year = ? AND semester = ?',

(student\_no, selected\_year, selected\_semester)

)

payments = self.cursor.fetchall()

for term, amount, date in payments:

payment\_data[term] = {"Amount": amount, "Date": date}

for term in self.payment\_terms:

pay\_info = payment\_data.get(term, {"Amount": 0.0, "Date": ""})

payment\_tree.insert('', 'end', iid=term, values=(term, f"{pay\_info.get('Amount',0):.2f}", pay\_info.get('Date','')))

# Remaining balance

total\_fee = student.get("tuition\_fee", 10000)

total\_paid = sum(p["Amount"] for p in payment\_data.values())

remaining = max(total\_fee - total\_paid, 0)

remaining\_balance\_var.set(f"Remaining Balance: ₱{remaining:.2f}")

remaining\_balance\_var = tk.StringVar()

lbl\_remaining = tk.Label(self.student\_right\_frame, textvariable=remaining\_balance\_var,

font=("Segoe UI", 16, "bold"), bg="#ffffff", fg="#dc2626")

lbl\_remaining.pack(side="bottom", pady=(10, 30), padx=60, anchor='e')

def on\_period\_change(\*args):

load\_payments\_for\_period()

year\_var.trace\_add('write', on\_period\_change)

semester\_var.trace\_add('write', on\_period\_change)

# Initial load

load\_payments\_for\_period()

def show\_student\_payment\_record(self, student\_no):##################################################

for widget in self.main\_frame.winfo\_children():

widget.destroy()

# Get student info and payments

student = next((s for s in self.students if s.get("student\_no") == student\_no), None)

if not student:

messagebox.showerror("Payment Record", "Student not found.")

self.load\_login\_view()

return

self.cursor.execute('SELECT term, amount, date FROM payments WHERE student\_no = ?', (student\_no,))

payments = self.cursor.fetchall()

payment\_data = {term: {"Amount": 0.0, "Date": ""} for term in self.payment\_terms}

for term, amount, date in payments:

payment\_data[term] = {"Amount": amount, "Date": date}

nav = tk.Frame(self.main\_frame, bg="#f3f4f6", height=48)

nav.pack(fill='x', side='top')

lbl\_brand = tk.Label(nav, text="Padre Garcia Polytechnic College", font=("Segoe UI", 16, "bold"), bg="#f3f4f6", fg="#1f2937")

lbl\_brand.pack(side='left', padx=24, pady=12)

btn\_back = ttk.Button(nav, text="Back", command=lambda: self.show\_student\_dashboard(student\_no), style="Danger.TButton")

btn\_back.pack(side='right', padx=24, pady=8)

center = tk.Frame(self.main\_frame, bg="#ffffff")

center.pack(expand=True, fill='both')

lbl\_title = tk.Label(center, text="Payment Record", font=("Segoe UI", 20, "bold"), bg="#ffffff", fg="#1f2937")

lbl\_title.pack(pady=(40, 20))

columns = ("Term", "Amount (₱)", "Date")

payment\_tree = ttk.Treeview(center, columns=columns, show="headings", height=12)

payment\_tree.pack(fill='both', expand=True, padx=60, pady=10)

for col in columns:

payment\_tree.heading(col, text=col)

payment\_tree.column(col, anchor='center')

for term in self.payment\_terms:

pay\_info = payment\_data.get(term, {"Amount": 0.0, "Date": ""})

payment\_tree.insert('', 'end', iid=term, values=(term, f"{pay\_info.get('Amount',0):.2f}", pay\_info.get('Date','')))

# Remaining balance

total\_fee = student.get("tuition\_fee", 10000)

total\_paid = sum(p["Amount"] for p in payment\_data.values())

remaining = max(total\_fee - total\_paid, 0)

remaining\_balance\_var = tk.StringVar(value=f"Remaining Balance: ₱{remaining:.2f}")

lbl\_remaining = tk.Label(center, textvariable=remaining\_balance\_var,

font=("Segoe UI", 16, "bold"), bg="#ffffff", fg="#dc2626")

lbl\_remaining.pack(side="bottom", pady=(10, 30), padx=60, anchor='e')

# ...existing code...

def load\_admin\_login\_panel(self):

self.current\_view = "admin\_login\_panel"

for widget in self.main\_frame.winfo\_children():

widget.destroy()

title\_font = font.Font(family="Segoe UI", size=28, weight="bold")

lbl\_title = tk.Label(self.main\_frame, text="Admin Login",

font=title\_font, bg="#ffffff", fg="#1f2937")

lbl\_title.pack(pady=(70, 50))

entry\_font = font.Font(family="Segoe UI", size=16)

frm\_inputs = tk.Frame(self.main\_frame, bg="#ffffff")

frm\_inputs.pack(pady=(0, 50), padx=250)

# Label above username entry

lbl\_username = tk.Label(frm\_inputs, text="Admin Username", font=entry\_font,

bg="#ffffff", anchor='w')

lbl\_username.pack(fill='x', pady=(0, 5))

self.entry\_username = ttk.Entry(frm\_inputs, font=entry\_font)

self.entry\_username.pack(fill='x', pady=(0, 15))

# Label above password entry

lbl\_password = tk.Label(frm\_inputs, text="Password", font=entry\_font,

bg="#ffffff", anchor='w')

lbl\_password.pack(fill='x', pady=(0, 5))

self.entry\_password = ttk.Entry(frm\_inputs, font=entry\_font, show="\*")

self.entry\_password.pack(fill='x', pady=(0, 20))

btn\_login = ttk.Button(frm\_inputs, text="Login", command=self.validate\_admin\_login)

btn\_login.config(style="Primary.TButton")

btn\_login.pack(fill='x')

btn\_back = ttk.Button(frm\_inputs, text="Back", command=self.load\_welcome\_view)

btn\_back.pack(fill='x', pady=(20, 0))

def validate\_admin\_login(self):

username = self.entry\_username.get()

password = self.entry\_password.get()

if username == "admin" and password == "password123":

messagebox.showinfo("Login Successful", "Welcome, Admin!")

self.load\_manage\_student\_panel() # Go directly to management panel

else:

messagebox.showerror("Login Failed", "Invalid Admin Username or Password.")

def load\_admin\_dashboard(self):

self.current\_view = "admin\_dashboard"

for widget in self.main\_frame.winfo\_children():

widget.destroy()

nav = tk.Frame(self.main\_frame, bg="#f3f4f6", height=48)

nav.pack(fill='x', side='top')

lbl\_brand = tk.Label(nav, text="Padre Garcia Polytechnic College", font=("Segoe UI", 16, "bold"), bg="#f3f4f6", fg="#1f2937")

lbl\_brand.pack(side='left', padx=24, pady=12)

btn\_logout = ttk.Button(nav, text="Logout", command=self.load\_welcome\_view, style="Danger.TButton")

btn\_logout.pack(side='right', padx=24, pady=8)

center = tk.Frame(self.main\_frame, bg="#ffffff")

center.pack(expand=True, fill='both')

lbl\_dashboard = tk.Label(center, text="Admin Dashboard", font=("Segoe UI", 22, "bold"), bg="#ffffff", fg="#1f2937")

lbl\_dashboard.pack(pady=(50, 10))

lbl\_info = tk.Label(center, text="Manage the system with the options below.", font=("Segoe UI", 12), bg="#ffffff", fg="#4b5563")

lbl\_info.pack(pady=(0, 30))

btn\_register = ttk.Button(center, text="Register Student", command=lambda: self.load\_student\_registration\_panel(from\_admin=True), style="Primary.TButton")

btn\_register.pack(ipadx=10, ipady=8, fill='x', padx=100, pady=(0, 15))

btn\_manage = ttk.Button(center, text="Manage Students", command=self.load\_manage\_student\_panel, style="Success.TButton")

btn\_manage.pack(ipadx=10, ipady=8, fill='x', padx=100, pady=(0, 15))

lbl\_footer = tk.Label(center, text="© 2025 Padre Garcia Polytechnic College", font=("Segoe UI", 10), bg="#ffffff", fg="#9ca3af")

lbl\_footer.pack(side="bottom", pady=(40, 0))

def load\_student\_registration\_panel(self, from\_admin=False):

self.current\_view = "register\_student"

for widget in self.main\_frame.winfo\_children():

widget.destroy()

nav = tk.Frame(self.main\_frame, bg="#f3f4f6", height=40)

nav.pack(fill='x', side='top')

lbl\_brand = tk.Label(nav, text="Padre Garcia Polytechnic College", font=("Segoe UI", 13, "bold"), bg="#f3f4f6", fg="#22223b")

lbl\_brand.pack(side='left', padx=12, pady=8)

# Context-aware back button

if from\_admin:

btn\_back = ttk.Button(nav, text="Back", command=self.load\_admin\_dashboard, style="Danger.TButton")

else:

btn\_back = ttk.Button(nav, text="Back", command=self.load\_student\_login\_panel, style="Danger.TButton")

btn\_back.pack(side='right', padx=12, pady=8)

center = tk.Frame(self.main\_frame, bg="#fff")

center.pack(expand=True, fill='both')

lbl\_title = tk.Label(center, text="Student Registration", font=("Segoe UI", 16, "bold"), bg="#fff", fg="#22223b")

lbl\_title.pack(pady=(20, 10))

entry\_font = font.Font(family="Segoe UI", size=11)

frm\_inputs = tk.Frame(center, bg="#fff", bd=1, relief="solid")

frm\_inputs.pack(pady=(0, 10), padx=120, fill="x")

# Last Name, First Name, Middle Name

frm\_names = tk.Frame(frm\_inputs, bg="#fff")

frm\_names.pack(fill='x', pady=(0, 5))

lbl\_last\_name = tk.Label(frm\_names, text="Last Name:", font=entry\_font, bg="#fff", anchor='w')

lbl\_last\_name.grid(row=0, column=0, sticky='w', padx=3)

self.entry\_last\_name = ttk.Entry(frm\_names, font=entry\_font, width=18)

self.entry\_last\_name.grid(row=1, column=0, sticky='ew', padx=3)

lbl\_first\_name = tk.Label(frm\_names, text="First Name:", font=entry\_font, bg="#fff", anchor='w')

lbl\_first\_name.grid(row=0, column=1, sticky='w', padx=3)

self.entry\_first\_name = ttk.Entry(frm\_names, font=entry\_font, width=18)

self.entry\_first\_name.grid(row=1, column=1, sticky='ew', padx=3)

lbl\_middle\_name = tk.Label(frm\_names, text="Middle Name:", font=entry\_font, bg="#fff", anchor='w')

lbl\_middle\_name.grid(row=0, column=2, sticky='w', padx=3)

self.entry\_middle\_name = ttk.Entry(frm\_names, font=entry\_font, width=18)

self.entry\_middle\_name.grid(row=1, column=2, sticky='ew', padx=3)

# Student Number and Password

frm\_credentials = tk.Frame(frm\_inputs, bg="#fff")

frm\_credentials.pack(fill='x', pady=(0, 5))

lbl\_student\_no = tk.Label(frm\_credentials, text="Student Number:", font=entry\_font, bg="#fff", anchor='w')

lbl\_student\_no.grid(row=0, column=0, sticky='w', padx=3)

self.entry\_student\_no = ttk.Entry(frm\_credentials, font=entry\_font)

self.entry\_student\_no.grid(row=1, column=0, sticky='ew', padx=3)

lbl\_password = tk.Label(frm\_credentials, text="Password:", font=entry\_font, bg="#fff", anchor='w')

lbl\_password.grid(row=0, column=1, sticky='w', padx=3)

self.entry\_password = ttk.Entry(frm\_credentials, font=entry\_font, show='\*')

self.entry\_password.grid(row=1, column=1, sticky='ew', padx=3)

# Contact Number and Gmail

frm\_contact = tk.Frame(frm\_inputs, bg="#fff")

frm\_contact.pack(fill='x', pady=(0, 5))

lbl\_contact = tk.Label(frm\_contact, text="Contact Number:", font=entry\_font, bg="#fff", anchor='w')

lbl\_contact.grid(row=0, column=0, sticky='w', padx=3)

self.entry\_contact = ttk.Entry(frm\_contact, font=entry\_font)

self.entry\_contact.grid(row=1, column=0, sticky='ew', padx=3)

lbl\_gmail = tk.Label(frm\_contact, text="Gmail:", font=entry\_font, bg="#fff", anchor='w')

lbl\_gmail.grid(row=0, column=1, sticky='w', padx=3)

self.entry\_gmail = ttk.Entry(frm\_contact, font=entry\_font)

self.entry\_gmail.grid(row=1, column=1, sticky='ew', padx=3)

# Course, Semester, Year, Section, Category

frm\_details = tk.Frame(frm\_inputs, bg="#fff")

frm\_details.pack(fill='x', pady=(0, 5))

lbl\_course = tk.Label(frm\_details, text="Course:", font=entry\_font, bg="#fff", anchor='w')

lbl\_course.grid(row=0, column=0, sticky='w', padx=3)

self.course\_var = tk.StringVar()

self.course\_dropdown = ttk.Combobox(frm\_details, textvariable=self.course\_var, font=entry\_font, state="readonly")

self.course\_dropdown['values'] = [

"Bachelor of Science in Computer Science",

"Bachelor of Science in Management Accounting",

"Bachelor of Science in Public Administration",

"Bachelor of Science in Criminology"

]

self.course\_dropdown.grid(row=1, column=0, sticky='ew', padx=3)

lbl\_semester = tk.Label(frm\_details, text="Semester:", font=entry\_font, bg="#fff", anchor='w')

lbl\_semester.grid(row=0, column=1, sticky='w', padx=3)

self.semester\_var = tk.StringVar()

self.semester\_dropdown = ttk.Combobox(frm\_details, textvariable=self.semester\_var, font=entry\_font, state="readonly")

self.semester\_dropdown['values'] = ["First Semester", "Second Semester"]

self.semester\_dropdown.grid(row=1, column=1, sticky='ew', padx=3)

lbl\_year = tk.Label(frm\_details, text="Year:", font=entry\_font, bg="#fff", anchor='w')

lbl\_year.grid(row=0, column=2, sticky='w', padx=3)

self.year\_var = tk.StringVar()

self.year\_dropdown = ttk.Combobox(frm\_details, textvariable=self.year\_var, font=entry\_font, state="readonly")

self.year\_dropdown['values'] = ["1st Year", "2nd Year", "3rd Year", "4th Year"]

self.year\_dropdown.grid(row=1, column=2, sticky='ew', padx=3)

self.year\_dropdown.bind("<<ComboboxSelected>>", self.update\_section\_options)

lbl\_section = tk.Label(frm\_details, text="Section:", font=entry\_font, bg="#fff", anchor='w')

lbl\_section.grid(row=0, column=3, sticky='w', padx=3)

self.section\_var = tk.StringVar()

self.section\_dropdown = ttk.Combobox(frm\_details, textvariable=self.section\_var, font=entry\_font, state="readonly")

self.section\_dropdown.grid(row=1, column=3, sticky='ew', padx=3)

lbl\_category = tk.Label(frm\_inputs, text="Category:", font=entry\_font, bg="#fff", anchor='w')

lbl\_category.pack(fill='x', padx=10, pady=(8, 2))

self.category\_var = tk.StringVar()

self.category\_dropdown = ttk.Combobox(frm\_inputs, textvariable=self.category\_var, font=entry\_font, state="readonly")

self.category\_dropdown['values'] = ["Garciano", "Non-Garciano"]

self.category\_dropdown.pack(fill='x', padx=10, pady=(8, 2))

btn\_register = ttk.Button(center, text="Register Student", command=self.register\_student, style="Primary.TButton")

btn\_register.pack(pady=(10, 0), padx=40, fill='x')

def update\_section\_options(self, event=None):

year = self.year\_var.get()

mapping = {

"1st Year": ["1A", "1B", "1C", "1D", "1E", "1F"],

"2nd Year": ["2A", "2B", "2C", "2D", "2E", "2F"],

"3rd Year": ["3A", "3B", "3C", "3D", "3E", "3F"],

"4th Year": ["4A", "4B", "4C", "4D", "4E", "4F"],

}

self.section\_dropdown['values'] = mapping.get(year, [])

if self.section\_dropdown['values']:

self.section\_var.set(self.section\_dropdown['values'][0])

else:

self.section\_var.set('')

def calculate\_tuition\_fee(self, course, category, semester):

try:

return self.tuition\_fees[category][semester][course]

except KeyError:

return 0

def register\_student(self):

student\_data = {

"last\_name": self.entry\_last\_name.get().strip(),

"first\_name": self.entry\_first\_name.get().strip(),

"middle\_name": self.entry\_middle\_name.get().strip(),

"student\_no": self.entry\_student\_no.get().strip(),

"password": self.entry\_password.get().strip(),

"contact\_number": self.entry\_contact.get().strip(),

"gmail": self.entry\_gmail.get().strip(),

"course": self.course\_var.get().strip(),

"section": self.section\_var.get().strip(),

"year": self.year\_var.get().strip(),

"semester": self.semester\_var.get().strip(),

"category": self.category\_var.get().strip()

}

required\_fields = ["last\_name", "first\_name", "student\_no", "password", "course", "section", "year", "semester", "category"]

for field in required\_fields:

if not student\_data[field]:

messagebox.showerror("Input Error", f"{field.replace('\_', ' ').title()} is required.")

return

self.cursor.execute('SELECT 1 FROM students WHERE student\_no = ?', (student\_data["student\_no"],))

if self.cursor.fetchone():

messagebox.showerror("Duplicate Entry", "Student number already registered.")

return

tuition\_fee = self.calculate\_tuition\_fee(student\_data["course"], student\_data["category"], student\_data["semester"])

student\_data["tuition\_fee"] = tuition\_fee

try:

self.cursor.execute('''

INSERT INTO students

(last\_name, first\_name, middle\_name, student\_no, password, contact\_number, gmail, course, section, year, semester, category, tuition\_fee)

VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)

''', (student\_data["last\_name"], student\_data["first\_name"], student\_data["middle\_name"], student\_data["student\_no"],

student\_data["password"], student\_data["contact\_number"], student\_data["gmail"], student\_data["course"], student\_data["section"],

student\_data["year"], student\_data["semester"], student\_data["category"], student\_data["tuition\_fee"]))

# Insert into enrollments table

self.cursor.execute('''

INSERT INTO enrollments (student\_no, year, semester, date\_enrolled)

VALUES (?, ?, ?, ?)

''', (

student\_data["student\_no"],

student\_data["year"],

student\_data["semester"],

datetime.date.today().isoformat()

))

self.conn.commit()

messagebox.showinfo("Success", f"Student registered successfully.\nTuition Fee: ₱{tuition\_fee:.2f}")

except sqlite3.Error as e:

messagebox.showerror("Database Error", f"An error occurred: {e}")

return

self.load\_students\_from\_db()

self.show\_manage\_student\_list()

def load\_manage\_student\_panel(self):

self.current\_view = "manage\_student"

for widget in self.main\_frame.winfo\_children():

widget.destroy()

fw = self.main\_frame.winfo\_width()

fh = self.main\_frame.winfo\_height()

btn\_frame\_width = max(300, int(fw \* 0.25))

self.btn\_frame = tk.Frame(self.main\_frame, bg="#f9fafb", width=btn\_frame\_width)

self.btn\_frame.pack(side="left", fill="y")

# Register Student button at the top

btn\_register = ttk.Button(self.btn\_frame, text="Register Student", command=self.show\_registration\_form, style="Sidebar.TButton")

btn\_register.pack(fill='x', padx=30, pady=(30, 10), ipady=10)

btn\_home = ttk.Button(self.btn\_frame, text="Refresh List", command=self.load\_manage\_student\_panel, style="Sidebar.TButton")

btn\_home.pack(fill='x', padx=30, pady=(10, 10), ipady=10)

btn\_edit = ttk.Button(self.btn\_frame, text="Edit Data", command=self.edit\_student\_data, style="Sidebar.TButton")

btn\_edit.pack(fill='x', padx=30, pady=10, ipady=10)

btn\_payment = ttk.Button(self.btn\_frame, text="Payment Record", command=self.payment\_record, style="Sidebar.TButton")

btn\_payment.pack(fill='x', padx=30, pady=10, ipady=10)

btn\_delete = ttk.Button(self.btn\_frame, text="Delete Account", command=self.delete\_account, style="Sidebar.TButton")

btn\_delete.pack(fill='x', padx=30, pady=10, ipady=10)

btn\_logout = ttk.Button(self.btn\_frame, text="Logout", command=self.load\_welcome\_view, style="Sidebar.TButton")

btn\_logout.pack(fill='x', padx=30, pady=(10, 20), ipady=10)

self.right\_frame = tk.Frame(self.main\_frame, bg="#ffffff")

self.right\_frame.pack(side="left", expand=True, fill="both")

# By default, show the student list

self.show\_manage\_student\_list()

def show\_registration\_form(self):

for widget in self.right\_frame.winfo\_children():

widget.destroy()

lbl\_title = tk.Label(self.right\_frame, text="Student Registration", font=("Segoe UI", 16, "bold"), bg="#fff", fg="#22223b")

lbl\_title.pack(pady=(20, 10))

entry\_font = font.Font(family="Segoe UI", size=11)

frm\_inputs = tk.Frame(self.right\_frame, bg="#fff", bd=1, relief="solid")

frm\_inputs.pack(pady=(0, 10), padx=120, fill="x")

# Last Name, First Name, Middle Name

frm\_names = tk.Frame(frm\_inputs, bg="#fff")

frm\_names.pack(fill='x', pady=(0, 5))

lbl\_last\_name = tk.Label(frm\_names, text="Last Name:", font=entry\_font, bg="#fff", anchor='w')

lbl\_last\_name.grid(row=0, column=0, sticky='w', padx=3)

self.entry\_last\_name = ttk.Entry(frm\_names, font=entry\_font, width=18)

self.entry\_last\_name.grid(row=1, column=0, sticky='ew', padx=3)

lbl\_first\_name = tk.Label(frm\_names, text="First Name:", font=entry\_font, bg="#fff", anchor='w')

lbl\_first\_name.grid(row=0, column=1, sticky='w', padx=3)

self.entry\_first\_name = ttk.Entry(frm\_names, font=entry\_font, width=18)

self.entry\_first\_name.grid(row=1, column=1, sticky='ew', padx=3)

lbl\_middle\_name = tk.Label(frm\_names, text="Middle Name:", font=entry\_font, bg="#fff", anchor='w')

lbl\_middle\_name.grid(row=0, column=2, sticky='w', padx=3)

self.entry\_middle\_name = ttk.Entry(frm\_names, font=entry\_font, width=18)

self.entry\_middle\_name.grid(row=1, column=2, sticky='ew', padx=3)

# Student Number and Password

frm\_credentials = tk.Frame(frm\_inputs, bg="#fff")

frm\_credentials.pack(fill='x', pady=(0, 5))

lbl\_student\_no = tk.Label(frm\_credentials, text="Student Number:", font=entry\_font, bg="#fff", anchor='w')

lbl\_student\_no.grid(row=0, column=0, sticky='w', padx=3)

self.entry\_student\_no = ttk.Entry(frm\_credentials, font=entry\_font)

self.entry\_student\_no.grid(row=1, column=0, sticky='ew', padx=3)

lbl\_password = tk.Label(frm\_credentials, text="Password:", font=entry\_font, bg="#fff", anchor='w')

lbl\_password.grid(row=0, column=1, sticky='w', padx=3)

self.entry\_password = ttk.Entry(frm\_credentials, font=entry\_font, show='\*')

self.entry\_password.grid(row=1, column=1, sticky='ew', padx=3)

# Contact Number and Gmail

frm\_contact = tk.Frame(frm\_inputs, bg="#fff")

frm\_contact.pack(fill='x', pady=(0, 5))

lbl\_contact = tk.Label(frm\_contact, text="Contact Number:", font=entry\_font, bg="#fff", anchor='w')

lbl\_contact.grid(row=0, column=0, sticky='w', padx=3)

self.entry\_contact = ttk.Entry(frm\_contact, font=entry\_font)

self.entry\_contact.grid(row=1, column=0, sticky='ew', padx=3)

lbl\_gmail = tk.Label(frm\_contact, text="Gmail:", font=entry\_font, bg="#fff", anchor='w')

lbl\_gmail.grid(row=0, column=1, sticky='w', padx=3)

self.entry\_gmail = ttk.Entry(frm\_contact, font=entry\_font)

self.entry\_gmail.grid(row=1, column=1, sticky='ew', padx=3)

# Course, Semester, Year, Section, Category

frm\_details = tk.Frame(frm\_inputs, bg="#fff")

frm\_details.pack(fill='x', pady=(0, 5))

lbl\_course = tk.Label(frm\_details, text="Course:", font=entry\_font, bg="#fff", anchor='w')

lbl\_course.grid(row=0, column=0, sticky='w', padx=3)

self.course\_var = tk.StringVar()

self.course\_dropdown = ttk.Combobox(frm\_details, textvariable=self.course\_var, font=entry\_font, state="readonly")

self.course\_dropdown['values'] = [

"Bachelor of Science in Computer Science",

"Bachelor of Science in Management Accounting",

"Bachelor of Science in Public Administration",

"Bachelor of Science in Criminology"

]

self.course\_dropdown.grid(row=1, column=0, sticky='ew', padx=3)

lbl\_semester = tk.Label(frm\_details, text="Semester:", font=entry\_font, bg="#fff", anchor='w')

lbl\_semester.grid(row=0, column=1, sticky='w', padx=3)

self.semester\_var = tk.StringVar()

self.semester\_dropdown = ttk.Combobox(frm\_details, textvariable=self.semester\_var, font=entry\_font, state="readonly")

self.semester\_dropdown['values'] = ["First Semester", "Second Semester"]

self.semester\_dropdown.grid(row=1, column=1, sticky='ew', padx=3)

lbl\_year = tk.Label(frm\_details, text="Year:", font=entry\_font, bg="#fff", anchor='w')

lbl\_year.grid(row=0, column=2, sticky='w', padx=3)

self.year\_var = tk.StringVar()

self.year\_dropdown = ttk.Combobox(frm\_details, textvariable=self.year\_var, font=entry\_font, state="readonly")

self.year\_dropdown['values'] = ["1st Year", "2nd Year", "3rd Year", "4th Year"]

self.year\_dropdown.grid(row=1, column=2, sticky='ew', padx=3)

self.year\_dropdown.bind("<<ComboboxSelected>>", self.update\_section\_options)

lbl\_section = tk.Label(frm\_details, text="Section:", font=entry\_font, bg="#fff", anchor='w')

lbl\_section.grid(row=0, column=3, sticky='w', padx=3)

self.section\_var = tk.StringVar()

self.section\_dropdown = ttk.Combobox(frm\_details, textvariable=self.section\_var, font=entry\_font, state="readonly")

self.section\_dropdown.grid(row=1, column=3, sticky='ew', padx=3)

lbl\_category = tk.Label(frm\_inputs, text="Category:", font=entry\_font, bg="#fff", anchor='w')

lbl\_category.pack(fill='x', padx=10, pady=(8, 2))

self.category\_var = tk.StringVar()

self.category\_dropdown = ttk.Combobox(frm\_inputs, textvariable=self.category\_var, font=entry\_font, state="readonly")

self.category\_dropdown['values'] = ["Garciano", "Non-Garciano"]

self.category\_dropdown.pack(fill='x', padx=10, pady=(8, 2))

btn\_register = ttk.Button(self.right\_frame, text="Register", command=self.register\_student, style="Primary.TButton")

btn\_register.pack(pady=(10, 0), padx=40, fill='x')

def show\_manage\_student\_list(self, course\_filter=None, search\_query=""):

for widget in self.right\_frame.winfo\_children():

widget.destroy()

# --- Add course filter buttons ---

filter\_frame = tk.Frame(self.right\_frame, bg="#ffffff")

filter\_frame.pack(fill='x', padx=30, pady=(20, 0))

courses = [

"All",

"Bachelor of Science in Computer Science",

"Bachelor of Science in Management Accounting",

"Bachelor of Science in Public Administration",

"Bachelor of Science in Criminology"

]

for course in courses:

btn = ttk.Button(

filter\_frame,

text=course if course != "All" else "All Courses",

style="Primary.TButton" if course\_filter == course or (course == "All" and not course\_filter) else "TButton",

command=lambda c=course: self.show\_manage\_student\_list(None if c == "All" else c)

)

btn.pack(side="left", padx=5)

# --- Search bar ---

search\_frame = tk.Frame(self.right\_frame, bg="#ffffff")

search\_frame.pack(fill='x', padx=30, pady=(10, 0))

tk.Label(search\_frame, text="Search:", bg="#ffffff", font=("Segoe UI", 11)).pack(side="left")

search\_var = tk.StringVar(value=search\_query)

search\_entry = ttk.Entry(search\_frame, textvariable=search\_var, width=30)

search\_entry.pack(side="left", padx=8)

def do\_search(event=None):

self.show\_manage\_student\_list(course\_filter, search\_var.get().strip())

search\_entry.bind("<Return>", do\_search)

ttk.Button(search\_frame, text="Go", command=do\_search, style="Primary.TButton").pack(side="left", padx=4)

# --- Title ---

title\_font = font.Font(family="Segoe UI", size=24, weight="bold")

lbl\_title = tk.Label(self.right\_frame, text="Student Info",

font=title\_font, bg="#ffffff", fg="#1f2937")

lbl\_title.pack(pady=(10, 20), padx=30, anchor='w')

cols = ("Student No", "Last Name", "First Name", "Middle Name", "Contact Number",

"Gmail", "Course", "Year", "Section", "Semester", "Category", "Tuition Fee")

# --- Treeview with Scrollbars ---

tree\_frame = tk.Frame(self.right\_frame, bg="#ffffff")

tree\_frame.pack(fill='both', expand=True, padx=30, pady=(0, 20))

self.tree = ttk.Treeview(tree\_frame, columns=cols, show="headings", height=20)

self.tree.pack(side="left", fill='both', expand=True)

# Vertical scrollbar

vsb = ttk.Scrollbar(tree\_frame, orient="vertical", command=self.tree.yview)

vsb.pack(side='right', fill='y')

self.tree.configure(yscrollcommand=vsb.set)

# Horizontal scrollbar

hsb = ttk.Scrollbar(tree\_frame, orient="horizontal", command=self.tree.xview)

hsb.pack(side='bottom', fill='x')

self.tree.configure(xscrollcommand=hsb.set)

# --- Filter students ---

filtered\_students = self.students

if course\_filter:

filtered\_students = [s for s in filtered\_students if s.get("course") == course\_filter]

if search\_query:

sq = search\_query.lower()

filtered\_students = [

s for s in filtered\_students

if sq in s.get("student\_no", "").lower()

or sq in s.get("last\_name", "").lower()

or sq in s.get("first\_name", "").lower()

or sq in s.get("middle\_name", "").lower()

or sq in s.get("contact\_number", "").lower()

or sq in s.get("gmail", "").lower()

or sq in s.get("course", "").lower()

or sq in s.get("year", "").lower()

or sq in s.get("section", "").lower()

or sq in s.get("semester", "").lower()

or sq in s.get("category", "").lower()

]

for col in cols:

self.tree.heading(col, text=col)

self.tree.column(col, anchor='center', width=120, stretch=True)

for student in filtered\_students:

self.tree.insert('', 'end', values=(

student.get("student\_no", ""),

student.get("last\_name", ""),

student.get("first\_name", ""),

student.get("middle\_name", ""),

student.get("contact\_number", ""),

student.get("gmail", ""),

student.get("course", ""),

student.get("year", ""),

student.get("section", ""),

student.get("semester", ""),

student.get("category", ""),

f"₱{student.get('tuition\_fee', 0):,.2f}"

))

# --- Auto-resize columns to fit content ---

import tkinter.font as tkFont

def auto\_resize\_columns(tree, cols):

font = tkFont.Font()

for col in cols:

max\_width = font.measure(col)

for item in tree.get\_children():

cell\_value = str(tree.set(item, col))

cell\_width = font.measure(cell\_value)

if cell\_width > max\_width:

max\_width = cell\_width

tree.column(col, width=max\_width + 20) # add some padding

auto\_resize\_columns(self.tree, cols)

# --- Mouse wheel scrolling (vertical and horizontal, faster and easier) ---

def \_on\_mousewheel(event):

lines = 5 # Increase for even faster scroll

# If can't scroll vertically, scroll horizontally

if self.tree.yview() == (0.0, 1.0):

self.tree.xview\_scroll(int(-1\*(event.delta/120))\*lines, "units")

else:

self.tree.yview\_scroll(int(-1\*(event.delta/120))\*lines, "units")

return "break"

def \_on\_shift\_mousewheel(event):

lines = 5

self.tree.xview\_scroll(int(-1\*(event.delta/120))\*lines, "units")

return "break"

self.tree.bind("<MouseWheel>", \_on\_mousewheel)

self.tree.bind("<Shift-MouseWheel>", \_on\_shift\_mousewheel)

def refresh\_student\_list(self):

for item in self.tree.get\_children():

self.tree.delete(item)

for student in self.students:

self.tree.insert('', 'end', values=(

student.get("student\_no", ""),

student.get("last\_name", ""),

student.get("first\_name", ""),

student.get("middle\_name", ""),

student.get("contact\_number", ""),

student.get("gmail", ""),

student.get("course", ""),

student.get("year", ""),

student.get("section", ""),

student.get("semester", ""),

student.get("category", "")

))

def on\_student\_select(self, event):

selected = self.tree.selection()

if not selected:

return

sel = selected[0]

values = self.tree.item(sel, 'values')

student\_no = values[0]

self.editing\_student\_no = student\_no

if self.current\_view == 'manage\_student\_payment':

self.load\_payment\_records\_for\_student(student\_no)

def edit\_student\_data(self):

if not hasattr(self, 'tree') or not self.tree.winfo\_exists():

messagebox.showwarning("Edit Data", "Student list is not available. Please refresh the panel.")

return

selected = self.tree.selection()

if not selected:

messagebox.showwarning("Edit Data", "Please select a student to edit.")

return

sel = selected[0]

values = self.tree.item(sel, 'values')

student\_no = values[0]

student = next((s for s in self.students if s.get("student\_no") == student\_no), None)

if not student:

messagebox.showerror("Edit Data", "Student not found.")

return

self.show\_edit\_student\_form(student)

# ...rest of your code...

def show\_edit\_student\_form(self, student):

for widget in self.right\_frame.winfo\_children():

widget.destroy()

lbl\_title = tk.Label(self.right\_frame, text="Edit Student", font=("Segoe UI", 16, "bold"), bg="#fff", fg="#22223b")

lbl\_title.pack(pady=(20, 10))

entry\_font = font.Font(family="Segoe UI", size=11)

frm\_inputs = tk.Frame(self.right\_frame, bg="#fff", bd=1, relief="solid")

frm\_inputs.pack(pady=(0, 10), padx=120, fill="x")

# Last Name, First Name, Middle Name

frm\_names = tk.Frame(frm\_inputs, bg="#fff")

frm\_names.pack(fill='x', pady=(0, 5))

lbl\_last\_name = tk.Label(frm\_names, text="Last Name:", font=entry\_font, bg="#fff", anchor='w')

lbl\_last\_name.grid(row=0, column=0, sticky='w', padx=3)

entry\_last\_name = ttk.Entry(frm\_names, font=entry\_font, width=18)

entry\_last\_name.grid(row=1, column=0, sticky='ew', padx=3)

entry\_last\_name.insert(0, student.get("last\_name", ""))

lbl\_first\_name = tk.Label(frm\_names, text="First Name:", font=entry\_font, bg="#fff", anchor='w')

lbl\_first\_name.grid(row=0, column=1, sticky='w', padx=3)

entry\_first\_name = ttk.Entry(frm\_names, font=entry\_font, width=18)

entry\_first\_name.grid(row=1, column=1, sticky='ew', padx=3)

entry\_first\_name.insert(0, student.get("first\_name", ""))

lbl\_middle\_name = tk.Label(frm\_names, text="Middle Name:", font=entry\_font, bg="#fff", anchor='w')

lbl\_middle\_name.grid(row=0, column=2, sticky='w', padx=3)

entry\_middle\_name = ttk.Entry(frm\_names, font=entry\_font, width=18)

entry\_middle\_name.grid(row=1, column=2, sticky='ew', padx=3)

entry\_middle\_name.insert(0, student.get("middle\_name", ""))

# Student Number (read-only)

frm\_credentials = tk.Frame(frm\_inputs, bg="#fff")

frm\_credentials.pack(fill='x', pady=(0, 5))

lbl\_student\_no = tk.Label(frm\_credentials, text="Student Number:", font=entry\_font, bg="#fff", anchor='w')

lbl\_student\_no.grid(row=0, column=0, sticky='w', padx=3)

entry\_student\_no = ttk.Entry(frm\_credentials, font=entry\_font)

entry\_student\_no.grid(row=1, column=0, sticky='ew', padx=3)

entry\_student\_no.insert(0, student.get("student\_no", ""))

entry\_student\_no.config(state='readonly')

lbl\_password = tk.Label(frm\_credentials, text="Password:", font=entry\_font, bg="#fff", anchor='w')

lbl\_password.grid(row=0, column=1, sticky='w', padx=3)

entry\_password = ttk.Entry(frm\_credentials, font=entry\_font, show='\*')

entry\_password.grid(row=1, column=1, sticky='ew', padx=3)

entry\_password.insert(0, student.get("password", ""))

# Contact Number and Gmail

frm\_contact = tk.Frame(frm\_inputs, bg="#fff")

frm\_contact.pack(fill='x', pady=(0, 5))

lbl\_contact = tk.Label(frm\_contact, text="Contact Number:", font=entry\_font, bg="#fff", anchor='w')

lbl\_contact.grid(row=0, column=0, sticky='w', padx=3)

entry\_contact = ttk.Entry(frm\_contact, font=entry\_font)

entry\_contact.grid(row=1, column=0, sticky='ew', padx=3)

entry\_contact.insert(0, student.get("contact\_number", ""))

lbl\_gmail = tk.Label(frm\_contact, text="Gmail:", font=entry\_font, bg="#fff", anchor='w')

lbl\_gmail.grid(row=0, column=1, sticky='w', padx=3)

entry\_gmail = ttk.Entry(frm\_contact, font=entry\_font)

entry\_gmail.grid(row=1, column=1, sticky='ew', padx=3)

entry\_gmail.insert(0, student.get("gmail", ""))

# Course, Semester, Year, Section, Category

frm\_details = tk.Frame(frm\_inputs, bg="#fff")

frm\_details.pack(fill='x', pady=(0, 5))

lbl\_course = tk.Label(frm\_details, text="Course:", font=entry\_font, bg="#fff", anchor='w')

lbl\_course.grid(row=0, column=0, sticky='w', padx=3)

course\_var = tk.StringVar(value=student.get("course", ""))

course\_dropdown = ttk.Combobox(frm\_details, textvariable=course\_var, font=entry\_font, state="readonly")

course\_dropdown['values'] = [

"Bachelor of Science in Computer Science",

"Bachelor of Science in Management Accounting",

"Bachelor of Science in Public Administration",

"Bachelor of Science in Criminology"

]

course\_dropdown.grid(row=1, column=0, sticky='ew', padx=3)

lbl\_semester = tk.Label(frm\_details, text="Semester:", font=entry\_font, bg="#fff", anchor='w')

lbl\_semester.grid(row=0, column=1, sticky='w', padx=3)

semester\_var = tk.StringVar(value=student.get("semester", ""))

semester\_dropdown = ttk.Combobox(frm\_details, textvariable=semester\_var, font=entry\_font, state="readonly")

semester\_dropdown['values'] = ["First Semester", "Second Semester"]

semester\_dropdown.grid(row=1, column=1, sticky='ew', padx=3)

# ...existing code...

lbl\_year = tk.Label(frm\_details, text="Year:", font=entry\_font, bg="#fff", anchor='w')

lbl\_year.grid(row=0, column=2, sticky='w', padx=3)

year\_var = tk.StringVar(value=student.get("year", ""))

year\_dropdown = ttk.Combobox(frm\_details, textvariable=year\_var, font=entry\_font, state="readonly")

year\_dropdown['values'] = ["1st Year", "2nd Year", "3rd Year", "4th Year"]

year\_dropdown.grid(row=1, column=2, sticky='ew', padx=3)

lbl\_section = tk.Label(frm\_details, text="Section:", font=entry\_font, bg="#fff", anchor='w')

lbl\_section.grid(row=0, column=3, sticky='w', padx=3)

section\_var = tk.StringVar(value=student.get("section", ""))

section\_dropdown = ttk.Combobox(frm\_details, textvariable=section\_var, font=entry\_font, state="readonly")

section\_dropdown.grid(row=1, column=3, sticky='ew', padx=3)

# --- Add this block right after creating section\_dropdown ---

def update\_section\_options\_edit(event=None):

year = year\_var.get()

mapping = {

"1st Year": ["1A", "1B", "1C", "1D", "1E", "1F"],

"2nd Year": ["2A", "2B", "2C", "2D", "2E", "2F"],

"3rd Year": ["3A", "3B", "3C", "3D", "3E", "3F"],

"4th Year": ["4A", "4B", "4C", "4D", "4E", "4F"],

}

section\_dropdown['values'] = mapping.get(year, [])

if section\_dropdown['values']:

section\_var.set(section\_dropdown['values'][0])

else:

section\_var.set('')

year\_dropdown.bind("<<ComboboxSelected>>", update\_section\_options\_edit)

# --- Call once to set initial value based on loaded student ---

update\_section\_options\_edit()

# --- End of added block ---

lbl\_category = tk.Label(frm\_inputs, text="Category:", font=entry\_font, bg="#fff", anchor='w')

lbl\_category.pack(fill='x', padx=10, pady=(8, 2))

category\_var = tk.StringVar(value=student.get("category", ""))

category\_dropdown = ttk.Combobox(frm\_inputs, textvariable=category\_var, font=entry\_font, state="readonly")

category\_dropdown['values'] = ["Garciano", "Non-Garciano"]

category\_dropdown.pack(fill='x', padx=10, pady=(8, 2))

def save\_changes():

updated\_data = {

"last\_name": entry\_last\_name.get().strip(),

"first\_name": entry\_first\_name.get().strip(),

"middle\_name": entry\_middle\_name.get().strip(),

"password": entry\_password.get().strip(),

"contact\_number": entry\_contact.get().strip(),

"gmail": entry\_gmail.get().strip(),

"course": course\_var.get().strip(),

"section": section\_var.get().strip(),

"year": year\_var.get().strip(),

"semester": semester\_var.get().strip(),

"category": category\_var.get().strip()

}

required\_fields = ["last\_name", "first\_name", "course", "section", "year", "semester", "category"]

for field in required\_fields:

if not updated\_data[field]:

messagebox.showerror("Input Error", f"{field.replace('\_', ' ').title()} is required.")

return

tuition\_fee = self.calculate\_tuition\_fee(updated\_data["course"], updated\_data["category"], updated\_data["semester"])

updated\_data["tuition\_fee"] = tuition\_fee

try:

# Get old year/semester for comparison

old\_year = student.get("year")

old\_semester = student.get("semester")

self.cursor.execute('''

UPDATE students SET

last\_name = ?, first\_name = ?, middle\_name = ?, password = ?, contact\_number = ?, gmail = ?,

course = ?, section = ?, year = ?, semester = ?, category = ?, tuition\_fee = ?

WHERE student\_no = ?

''', (

updated\_data["last\_name"], updated\_data["first\_name"], updated\_data["middle\_name"], updated\_data["password"],

updated\_data["contact\_number"], updated\_data["gmail"], updated\_data["course"], updated\_data["section"],

updated\_data["year"], updated\_data["semester"], updated\_data["category"], updated\_data["tuition\_fee"],

student.get("student\_no")

))

# If year or semester changed, insert into enrollments

if old\_year != updated\_data["year"] or old\_semester != updated\_data["semester"]:

self.cursor.execute('''

INSERT INTO enrollments (student\_no, year, semester, date\_enrolled)

VALUES (?, ?, ?, ?)

''', (

student.get("student\_no"),

updated\_data["year"],

updated\_data["semester"],

datetime.date.today().isoformat()

))

self.conn.commit()

messagebox.showinfo("Success", "Student data updated successfully.")

self.load\_students\_from\_db()

self.show\_manage\_student\_list()

except sqlite3.Error as e:

messagebox.showerror("Database Error", f"An error occurred: {e}")

btn\_save = ttk.Button(self.right\_frame, text="Save Changes", command=save\_changes, style="Primary.TButton")

btn\_save.pack(pady=(10, 0), padx=40, fill='x')

def payment\_record(self):

selected = self.tree.selection()

if not selected:

messagebox.showwarning("Payment Record", "Please select a student to view payment records.")

return

sel = selected[0]

values = self.tree.item(sel, 'values')

student\_no = values[0]

self.editing\_student\_no = student\_no

self.load\_payment\_records\_for\_student(student\_no)

def load\_payment\_records\_for\_student(self, student\_no):

self.payment\_data = {term: {"Amount": 0.0, "Date": ""} for term in self.payment\_terms}

self.current\_view = "manage\_student\_payment"

for widget in self.right\_frame.winfo\_children():

widget.destroy()

import tkinter.font as tkFont

title\_font = tkFont.Font(family="Segoe UI", size=24, weight="bold")

student = next((s for s in self.students if s.get("student\_no") == student\_no), None)

student\_name = f"{student.get('first\_name','')} {student.get('last\_name','')}" if student else "Student"

lbl\_title = tk.Label(self.right\_frame, text=f"Payment Record for {student\_name}",

font=title\_font, bg="#ffffff", fg="#1f2937")

lbl\_title.pack(pady=(30, 10), padx=30, anchor='w')

# Dropdowns for year and semester

year\_options = ["1st Year", "2nd Year", "3rd Year", "4th Year"]

semester\_options = ["First Semester", "Second Semester"]

year\_var = tk.StringVar(value=student.get("year", year\_options[0]))

semester\_var = tk.StringVar(value=student.get("semester", semester\_options[0]))

select\_frame = tk.Frame(self.right\_frame, bg="#ffffff")

select\_frame.pack(pady=(0, 10), padx=60, anchor='w')

tk.Label(select\_frame, text="Year:", bg="#ffffff").pack(side="left")

year\_dropdown = ttk.Combobox(select\_frame, textvariable=year\_var, values=year\_options, state="readonly", width=12)

year\_dropdown.pack(side="left", padx=(5, 20))

tk.Label(select\_frame, text="Semester:", bg="#ffffff").pack(side="left")

semester\_dropdown = ttk.Combobox(select\_frame, textvariable=semester\_var, values=semester\_options, state="readonly", width=16)

semester\_dropdown.pack(side="left", padx=(5, 0))

columns = ("Term", "Amount (₱)", "Date")

self.payment\_tree = ttk.Treeview(self.right\_frame, columns=columns, show="headings", height=12)

self.payment\_tree.pack(fill='both', expand=True, padx=60, pady=10)

for col in columns:

self.payment\_tree.heading(col, text=col)

self.payment\_tree.column(col, anchor='center')

def load\_payments\_for\_period():

# Clear previous rows

for row in self.payment\_tree.get\_children():

self.payment\_tree.delete(row)

selected\_year = year\_var.get()

selected\_semester = semester\_var.get()

# 1. Initialize payment\_data with all terms

self.payment\_data = {term: {"Amount": 0.0, "Date": ""} for term in self.payment\_terms}

# 2. Fetch payments for the selected year/semester

self.cursor.execute(

'SELECT term, amount, date FROM payments WHERE student\_no = ? AND year = ? AND semester = ?',

(student\_no, selected\_year, selected\_semester)

)

payments = self.cursor.fetchall()

for term, amount, date in payments:

self.payment\_data[term] = {"Amount": amount, "Date": date}

# 3. Insert all terms as rows

for term in self.payment\_terms:

pay\_info = self.payment\_data.get(term, {"Amount": 0.0, "Date": ""})

self.payment\_tree.insert(

'', 'end', iid=term,

values=(term, f"{pay\_info.get('Amount', 0):.2f}", pay\_info.get('Date', ''))

)

# Update total fee for this period

self.total\_fee = self.calculate\_tuition\_fee(

student.get("course", ""),

student.get("category", ""),

selected\_semester

)

self.calculate\_remaining\_balance()

# Entry fields for editing (only Amount)

entry\_frame = tk.Frame(self.right\_frame, bg="#ffffff")

entry\_frame.pack(pady=(10, 0), padx=60, anchor='w')

tk.Label(entry\_frame, text="Amount (₱):", bg="#ffffff").grid(row=0, column=0, padx=5, pady=2)

self.entry\_amount = ttk.Entry(entry\_frame, width=15)

self.entry\_amount.grid(row=0, column=1, padx=5, pady=2)

self.selected\_term\_var = tk.StringVar()

self.enable\_inputs\_admin(False)

def on\_row\_select(event):

selected = self.payment\_tree.selection()

if not selected:

self.selected\_term\_var.set("")

self.enable\_inputs\_admin(False)

self.entry\_amount.delete(0, 'end')

return

term = selected[0]

self.selected\_term\_var.set(term)

self.enable\_inputs\_admin(True)

pdata = self.payment\_data.get(term, {})

self.entry\_amount.delete(0, 'end')

self.entry\_amount.insert(0, f"{pdata.get('Amount', 0):.2f}")

self.payment\_tree.bind("<<TreeviewSelect>>", on\_row\_select)

def update\_selected\_payment():

term = self.selected\_term\_var.get()

if term == "":

messagebox.showwarning("Update Payment", "Please select a payment term to update.")

return

try:

amount = float(self.entry\_amount.get())

if amount < 0:

raise ValueError

except ValueError:

messagebox.showerror("Invalid Input", "Please enter a non-negative valid amount.")

return

# Automatically use today's date

import datetime

date = datetime.date.today().isoformat()

# Calculate total paid if this update goes through

current\_paid = sum(p["Amount"] for t, p in self.payment\_data.items() if t != term)

new\_total\_paid = current\_paid + amount

if new\_total\_paid > self.total\_fee:

messagebox.showerror("Overpayment", f"Total payments (₱{new\_total\_paid:.2f}) exceed tuition fee (₱{self.total\_fee:.2f}).")

return

selected\_year = year\_var.get()

selected\_semester = semester\_var.get()

try:

self.cursor.execute(

'SELECT id FROM payments WHERE student\_no = ? AND term = ? AND year = ? AND semester = ?',

(student\_no, term, selected\_year, selected\_semester)

)

existing = self.cursor.fetchone()

if existing:

self.cursor.execute(

'UPDATE payments SET amount = ?, date = ? WHERE id = ?',

(amount, date, existing[0])

)

else:

self.cursor.execute(

'INSERT INTO payments (student\_no, term, amount, date, year, semester) VALUES (?, ?, ?, ?, ?, ?)',

(student\_no, term, amount, date, selected\_year, selected\_semester)

)

self.conn.commit()

except sqlite3.Error as e:

messagebox.showerror("Database Error", f"An error occurred while updating payments: {e}")

return

self.payment\_data[term] = {"Amount": amount, "Date": date}

self.payment\_tree.item(term, values=(term, f"{amount:.2f}", date))

self.calculate\_remaining\_balance()

messagebox.showinfo("Payment Updated", f"Payment info for '{term}' updated successfully.")

def delete\_selected\_payment():

term = self.selected\_term\_var.get()

if term == "":

messagebox.showwarning("Delete Payment", "Please select a payment term to delete.")

return

selected\_year = year\_var.get()

selected\_semester = semester\_var.get()

try:

self.cursor.execute(

'DELETE FROM payments WHERE student\_no = ? AND term = ? AND year = ? AND semester = ?',

(student\_no, term, selected\_year, selected\_semester)

)

self.conn.commit()

except sqlite3.Error as e:

messagebox.showerror("Database Error", f"An error occurred while deleting payment: {e}")

return

self.payment\_data[term] = {"Amount": 0.0, "Date": ""}

self.payment\_tree.item(term, values=(term, "0.00", ""))

self.calculate\_remaining\_balance()

messagebox.showinfo("Payment Deleted", f"Payment for '{term}' deleted.")

btn\_update = ttk.Button(entry\_frame, text="Update Payment", command=update\_selected\_payment)

btn\_update.grid(row=0, column=2, padx=10)

btn\_delete = ttk.Button(entry\_frame, text="Delete Payment", command=delete\_selected\_payment)

btn\_delete.grid(row=0, column=3, padx=10)

# Remaining balance

self.total\_fee = self.calculate\_tuition\_fee(

student.get("course", ""),

student.get("category", ""),

semester\_var.get()

)

self.remaining\_balance\_var = tk.StringVar()

self.calculate\_remaining\_balance()

self.lbl\_remaining = tk.Label(self.right\_frame, textvariable=self.remaining\_balance\_var,

font=("Segoe UI", 16, "bold"), bg="#ffffff", fg="#dc2626")

self.lbl\_remaining.pack(side="bottom", pady=(10, 30), padx=60, anchor='e')

# When year/semester changes, reload payments

def on\_period\_change(\*args):

load\_payments\_for\_period()

self.selected\_term\_var.set("")

self.enable\_inputs\_admin(False)

self.entry\_amount.delete(0, 'end')

year\_var.trace\_add('write', on\_period\_change)

semester\_var.trace\_add('write', on\_period\_change)

# Initial load

load\_payments\_for\_period()

# Add this helper for admin input enabling#############################################

def enable\_inputs\_admin(self, enable=True):

state = 'normal' if enable else 'disabled'

self.entry\_amount.config(state=state)

def on\_payment\_row\_select(self, event):

selected = self.payment\_tree.selection()

if not selected:

self.selected\_term\_var.set("")

self.enable\_inputs(False)

self.entry\_amount.delete(0, 'end')

self.entry\_date.delete(0, 'end')

return

term = selected[0]

self.selected\_term\_var.set(term)

self.enable\_inputs(True)

pdata = self.payment\_data.get(term, {})

self.entry\_amount.delete(0, 'end')

self.entry\_amount.insert(0, f"{pdata.get('Amount', 0):.2f}")

self.entry\_date.delete(0, 'end')

self.entry\_date.insert(0, pdata.get('Date', ''))

def update\_selected\_payment(self):

term = self.selected\_term\_var.get()

if term == "":

messagebox.showwarning("Update Payment", "Please select a payment term to update.")

return

try:

amount = float(self.entry\_amount.get())

if amount < 0:

raise ValueError

except ValueError:

messagebox.showerror("Invalid Input", "Please enter a non-negative valid amount.")

return

import datetime

date = datetime.date.today().isoformat()

if date == "":

messagebox.showerror("Invalid Input", "Please enter a date.")

return

try:

self.cursor.execute('SELECT id FROM payments WHERE student\_no = ? AND term = ?', (self.editing\_student\_no, term))

existing = self.cursor.fetchone()

if existing:

self.cursor.execute('UPDATE payments SET amount = ?, date = ? WHERE id = ?', (amount, date, existing[0]))

else:

self.cursor.execute('INSERT INTO payments (student\_no, term, amount, date) VALUES (?, ?, ?, ?)',

(self.editing\_student\_no, term, amount, date))

self.conn.commit()

except sqlite3.Error as e:

messagebox.showerror("Database Error", f"An error occurred while updating payments: {e}")

return

self.payment\_data[term] = {"Amount": amount, "Date": date}

self.payment\_tree.item(term, values=(term, f"{amount:.2f}", date))

self.calculate\_remaining\_balance()

messagebox.showinfo("Payment Updated", f"Payment info for '{term}' updated successfully.")

def calculate\_remaining\_balance(self):

total\_paid = sum(p["Amount"] for p in self.payment\_data.values())

remaining = max(self.total\_fee - total\_paid, 0)

self.remaining\_balance\_var.set(f"Remaining Balance: ₱{remaining:.2f}")

def delete\_account(self):

selected = self.tree.selection()

if not selected:

messagebox.showwarning("Delete Account", "Please select a student to delete.")

return

confirm = messagebox.askyesno("Delete Account", "Are you sure you want to delete the selected student(s)?")

if confirm:

for sel in selected:

values = self.tree.item(sel, 'values')

student\_no = values[0]

try:

self.cursor.execute('DELETE FROM payments WHERE student\_no = ?', (student\_no,))

self.cursor.execute('DELETE FROM students WHERE student\_no = ?', (student\_no,))

self.conn.commit()

except sqlite3.Error as e:

messagebox.showerror("Database Error", f"An error occurred while deleting: {e}")

return

self.load\_students\_from\_db()

self.refresh\_student\_list()

messagebox.showinfo("Deleted", "Selected student(s) have been deleted.")

def \_\_del\_\_(self):

try:

self.conn.close()

except Exception:

pass

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

root = tk.Tk()

app = PadreGarciaApp(root)

root.mainloop()