from tkinter import \*

from tkinter import ttk

from PIL import Image, ImageTk

from tkinter import messagebox

import mysql.connector

class Employee :

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.geometry("1530x790+0+0")

        self.root.title("ATTENDANCE MANAGEMENT SYSTEM")

        #Variable

        self.var\_department=StringVar()

        self.var\_position=StringVar()

        self.var\_year=StringVar()

        self.var\_shift=StringVar()

        self.var\_employee\_id =StringVar()

        self.var\_employee\_name=StringVar()

        self.var\_gender =StringVar()

        self.var\_product\_category=StringVar()

        self.var\_phone =StringVar()

        self.var\_email=StringVar()

        self.var\_address=StringVar()

        self.var\_verification\_id =StringVar()

         # Set background color

        self.root.configure(bg="thistle")

        # First Image

        img = Image.open(r"C:\Users\91639\OneDrive\Desktop\attendance system.png")

        img = img.resize((500, 130), Image.BICUBIC)

        self.photoimg = ImageTk.PhotoImage(img)

        f\_lbl = Label(self.root, image=self.photoimg)

        f\_lbl.grid(row=0, column=0, padx=10, pady=10)

        # Second Image

        img1= Image.open(r"C:\Users\91639\OneDrive\Desktop\simple.png")

        img1 = img1.resize((500, 130), Image.BICUBIC)

        self.photoimg1 = ImageTk.PhotoImage(img1)

        f\_lbl = Label(self.root, image=self.photoimg1)

        f\_lbl.grid(row=0, column=1, padx=10, pady=10)

        # Third Image

        img2 = Image.open(r"C:\Users\91639\OneDrive\Desktop\attendance system.png")

        img2 = img2.resize((500, 130), Image.BICUBIC)

        self.photoimg2 = ImageTk.PhotoImage(img2)

        f\_lbl = Label(self.root, image=self.photoimg2)

        f\_lbl.grid(row=0, column=2, padx=10, pady=10)

         # Set background color

        self.root.configure(bg="thistle")

        title\_label = Label(self.root, text="Employee Attendance", font=("times new roman", 35, "italic"), bg="white", fg="red")

        title\_label.grid(row=1, column=0, columnspan=3, pady=10)

     # Frame

        main\_frame = Frame(self.root,bd=2)

        main\_frame.place(x=6,y=230,width=15550,height=600)

    # Left Frame

        left\_frame = LabelFrame(main\_frame,bd=2,relief=RIDGE,text = "Employee Details",font=("times new roman",12,"bold" ))

        left\_frame.place(x=10,y=10,width = 710,height=535)

    # left Image

        img\_left  = Image.open(r"C:\Users\91639\OneDrive\Desktop\attendance system.png")

        img\_left = img\_left.resize((500, 130), Image.BICUBIC)

        self.photoimg\_left = ImageTk.PhotoImage(img\_left)

        f\_lbl = Label(self.root, image=self.photoimg\_left)

        f\_lbl.place(x=70,y=270,width=600,height=130)

    #work Information

        current\_course\_frame = LabelFrame(left\_frame,bd=2,relief=RIDGE,text = "Work Details",font=("times new roman",12,"bold" ))

        current\_course\_frame.place(x=5,y=150,width = 680,height=105)

    #Department

        dep\_label = Label(current\_course\_frame, text ="Department",font=("times new roman",12,"bold"))

        dep\_label.grid(row=0,column =0,padx=10)

        dep\_combo = ttk.Combobox(current\_course\_frame,textvariable=self.var\_department,font=("times new roman",12,"bold"), state = "readonly",width = 17)

        dep\_combo["values"] =("Select Department","Finance and Accounting","Operations","Sales and Marketing","Customer Service","Research and Development","Administration","Project Management","Business Development")

        dep\_combo.current(0)

        dep\_combo.grid(row=0,column=1,padx =2,pady=10,sticky=W)

    #Position

        pos\_label = Label(current\_course\_frame, text ="Position",font=("times new roman",12,"bold"))

        pos\_label.grid(row=0,column =2,padx=10)

        pos\_combo = ttk.Combobox(current\_course\_frame,textvariable=self.var\_position,font=("times new roman",12,"bold"), state = "readonly",width = 17)

        pos\_combo["values"] =("Select Position","Chief Executive Officer","Chief Technology Office","Operations Manager","Sales Manager","Software Engineer/Developer","Administrative Assistant","Graphic Designer","Financial Analyst")

        pos\_combo.current(0)

        pos\_combo.grid(row=0,column=3,padx =2,pady=10,sticky=W)

    #Year

        ye\_label = Label(current\_course\_frame, text ="Year",font=("times new roman",12,"bold"))

        ye\_label.grid(row=1,column =0,padx=10)

        ye\_combo = ttk.Combobox(current\_course\_frame,textvariable=self.var\_year,font=("times new roman",12,"bold"), state = "readonly",width = 17)

        ye\_combo["values"] =("Select Year","2020-2021","2021-2022","2022-2023","2023-2024")

        ye\_combo.current(0)

        ye\_combo.grid(row=1,column=1,padx =2,pady=10,sticky=W)

    #Shift

        shi\_label = Label(current\_course\_frame, text ="Shift",font=("times new roman",12,"bold"))

        shi\_label.grid(row=1,column =2,padx=10)

        shi\_combo = ttk.Combobox(current\_course\_frame,textvariable=self.var\_shift,font=("times new roman",12,"bold"), state = "readonly",width = 17)

        shi\_combo["values"] =("Select Shift","Day","Night")

        shi\_combo.current(0)

        shi\_combo.grid(row=1,column=3,padx =2,pady=10,sticky=W)

    #Employee Information

        Employee\_course\_frame = LabelFrame(left\_frame,bd=2,relief=RIDGE,text = "Employee Details",font=("times new roman",12,"bold" ))

        Employee\_course\_frame.place(x=5,y=260,width = 680,height=180)

    #Employee\_ID

        Employee\_ID\_label = Label(Employee\_course\_frame, text ="Employee ID :",font=("times new roman",12,"bold"))

        Employee\_ID\_label.grid(row=0,column =0,padx=10,sticky=W)

        Employee\_ID\_entry = ttk.Entry(Employee\_course\_frame,textvariable=self.var\_employee\_id,width=20,font=("times new roman",12,"bold"))

        Employee\_ID\_entry.grid(row = 0,column=2,padx=10,pady=5,sticky=W)

    #Employee Name

        Employee\_Name\_label = Label(Employee\_course\_frame, text ="Employee Name :",font=("times new roman",12,"bold"))

        Employee\_Name\_label.grid(row=0,column =3,pady=5,padx=10,sticky=W)

        Employee\_Name\_entry = ttk.Entry(Employee\_course\_frame,textvariable=self.var\_employee\_name,width=20,font=("times new roman",12,"bold"))

        Employee\_Name\_entry.grid(row = 0,column=4,padx=10,pady=5,sticky=W)

    #Gender

        Gender\_label = Label(Employee\_course\_frame, text ="Gender :",font=("times new roman",12,"bold"))

        Gender\_label.grid(row=1,column =0,pady=5,padx=10,sticky=W)

        #Gender\_entry = ttk.Entry(Employee\_course\_frame,textvariable=self.var\_gender,width=20,font=("times new roman",12,"bold"))

        #Gender\_entry.grid(row = 1,column=2,padx=10,pady=5,sticky=W)

        gender\_combo = ttk.Combobox(Employee\_course\_frame,textvariable=self.var\_gender,font=("times new roman",12,"bold"), state = "readonly",width = 17)

        gender\_combo["values"] =("Select Gender","Male(M)","Female(F)","Other(O)")

        gender\_combo.current(0)

        gender\_combo.grid(row=1,column=2,padx =2,pady=10,sticky=W)

    #Product Category

        Product\_label = Label(Employee\_course\_frame, text ="Product Category :",font=("times new roman",12,"bold"))

        Product\_label.grid(row=1,column =3,pady=5,padx=10,sticky=W)

        Product\_entry  = ttk.Entry(Employee\_course\_frame,textvariable=self.var\_product\_category,width=20,font=("times new roman",12,"bold"))

        Product\_entry.grid(row = 1,column=4,padx=10,pady=5,sticky=W)

    #Phone

        Phone\_label= Label(Employee\_course\_frame, text ="Phone Number :",font=("times new roman",12,"bold"))

        Phone\_label.grid(row=2,column =0,pady=5,padx=10,sticky=W)

        Phone\_entry = ttk.Entry(Employee\_course\_frame,textvariable=self.var\_phone,width=20,font=("times new roman",12,"bold"))

        Phone\_entry.grid(row = 2,column=2,padx=10,pady=5,sticky=W)

    #Email

        Email\_label = Label(Employee\_course\_frame, text ="Email :",font=("times new roman",12,"bold"))

        Email\_label.grid(row=2,column =3,pady=5,padx=10,sticky=W)

        Email\_entry = ttk.Entry(Employee\_course\_frame,textvariable=self.var\_email,width=20,font=("times new roman",12,"bold"))

        Email\_entry.grid(row = 2,column=4,padx=10,pady=5,sticky=W)

    #Address

        Address\_label = Label(Employee\_course\_frame, text ="Address :",font=("times new roman",12,"bold"))

        Address\_label.grid(row=3,column =0,pady=5,padx=10,sticky=W)

        Address\_entry = ttk.Entry(Employee\_course\_frame,textvariable=self.var\_address,width=20,font=("times new roman",12,"bold"))

        Address\_entry.grid(row = 3,column=2,padx=10,pady=5,sticky=W)

    #Verification\_ID

        Verification\_label = Label(Employee\_course\_frame, text ="Verification ID :",font=("times new roman",12,"bold"))

        Verification\_label.grid(row=3,column =3,pady=5,padx=10,sticky=W)

        Verification\_entry = ttk.Entry(Employee\_course\_frame,textvariable=self.var\_verification\_id,width=20,font=("times new roman",12,"bold"))

        Verification\_entry.grid(row = 3,column=4,padx=10,pady=5,sticky=W)

    #Radio\_Button

    #Button\_Frame

        # Button\_Frame

        btn\_frame = Frame(left\_frame, bd=2, relief=RIDGE,bg="white")

        btn\_frame.place(x=5, y=450, width=685, height=37)

        save\_btn = Button(btn\_frame, text="Save", command=self.add\_data,width=16, font=("times new roman", 13, "bold"), bg="light coral", fg="white")

        save\_btn.grid(row=0, column=0)

        update\_btn = Button(btn\_frame, text="Take Photo Sample", width=16,  font=("times new roman", 13, "bold"), bg="light coral", fg="white")

        update\_btn.grid(row=0, column=1)

        delete\_btn = Button(btn\_frame, text="Delete",command=self.delete\_data, width=16, font=("times new roman", 13, "bold"), bg="light coral", fg="white")

        delete\_btn.grid(row=0, column=2)

        reset\_btn = Button(btn\_frame, text="Updtae Photo Sample", width=16,  font=("times new roman", 13, "bold"), bg="light coral", fg="white")

        reset\_btn.grid(row=0, column=3)

    # Right Frame

        Right\_frame = LabelFrame(main\_frame,bd=2,relief=RIDGE,text = "Employee Details",font=("times new roman",12,"bold" ))

        Right\_frame.place(x=775,y=10,width=740,height=535)

    #Right Image

        img\_right  = Image.open(r"C:\Users\91639\OneDrive\Desktop\attendance system.png")

        img\_right = img\_left.resize((500, 130), Image.BICUBIC)

        self.photoimg\_right = ImageTk.PhotoImage(img\_right)

        f\_lbl = Label(self.root, image=self.photoimg\_right)

        f\_lbl.place(x=840,y=270,width=600,height=130)

        #Search System

        Search\_frame = LabelFrame(Right\_frame,bd=2,relief=RIDGE,text = "Search System",font=("times new roman",12,"bold" ))

        Search\_frame.place(x=5,y=140,width = 730,height=70)

        search\_label= Label(Search\_frame, text =" Search By:",font=("times new roman",13,"bold"))

        search\_label.grid(row=0,column =0,pady=5,padx=10,sticky=W)

        search\_combo = ttk.Combobox(Search\_frame,font=("times new roman",12,"bold"), state = "readonly",width = 17)

        search\_combo["values"] =("Select","Employee ID","Employee Name","Email")

        search\_combo.current(0)

        search\_combo.grid(row=0,column=1,padx =2,pady=10,sticky=W)

        search\_entry = ttk.Entry(Search\_frame,width=20,font=("times new roman",12,"bold"))

        search\_entry.grid(row =0,column=2,padx=10,pady=5,sticky=W)

        search\_btn = Button(Search\_frame, text="Search", width=12, font=("times new roman", 13, "bold"), bg="light coral", fg="white")

        search\_btn.grid(row=0, column=3,padx=2)

        show\_btn = Button(Search\_frame, text="Show All", width=12,  font=("times new roman", 13, "bold"), bg="light coral", fg="white")

        show\_btn.grid(row=0, column=4,padx=2)

        #Table Frame

        table\_frame = Frame(Right\_frame,bd=2,relief=RIDGE)

        table\_frame.place(x=5,y=220,width = 720,height=290)

        scroll\_x = ttk.Scrollbar(table\_frame,orient=HORIZONTAL)

        scroll\_y = ttk.Scrollbar(table\_frame,orient=VERTICAL)

        self.employee\_table = ttk.Treeview(table\_frame,column=("Employee\_ID","Employee\_Name","Gender","Product Category","Phone","Email","Address","Verification\_ID"),xscrollcommand=scroll\_x.set,yscrollcommand=scroll\_y.set)

        scroll\_x.pack(side=BOTTOM,fill=X)

        scroll\_y.pack(side=RIGHT,fill=Y)

        scroll\_x.config(command=self.employee\_table.xview)

        scroll\_y.config(command=self.employee\_table.yview)

        self.employee\_table["columns"] = ("Employee\_ID", "Employee\_Name", "Gender", "Product\_Category", "Phone", "Email", "Address", "Verification\_ID")

        self.employee\_table.column("#0", width=80)  # Adding a column for treeview hierarchy

        self.employee\_table.column("Employee\_ID", width=100)

        self.employee\_table.column("Employee\_Name", width=150)

        self.employee\_table.column("Gender", width=80)

        self.employee\_table.column("Product\_Category", width=150)

        self.employee\_table.column("Phone", width=120)

        self.employee\_table.column("Email", width=200)

        self.employee\_table.column("Address", width=200)

        self.employee\_table.column("Verification\_ID", width=120)

        self.employee\_table.heading("#0", text="ID")

        self.employee\_table.heading("Employee\_ID", text="Employee ID")

        self.employee\_table.heading("Employee\_Name", text="Employee Name")

        self.employee\_table.heading("Gender", text="Gender")

        self.employee\_table.heading("Product\_Category", text="Product Category")

        self.employee\_table.heading("Phone", text="Phone")

        self.employee\_table.heading("Email", text="Email")

        self.employee\_table.heading("Address", text="Address")

        self.employee\_table.heading("Verification\_ID", text="Verification ID")

        self.employee\_table["show"] = "headings"

        #self.employee\_table.pack(fill=BOTH,expand=1)

        #self.fetch\_data()

        self.employee\_table.pack(fill=BOTH, expand=1)

        self.employee\_table.bind("<ButtonRelease")

# Call fetch\_data method to populate the table initially

        self.fetch\_data()

#Function Decration

    def add\_data(self):

     if self.var\_department.get() == "Select Department" or self.var\_employee\_name.get() == "" or self.var\_employee\_id.get() == "":

        messagebox.showerror("Error", "All fields are required", parent=self.root)

     else:

        try:

            conn = mysql.connector.connect(host="localhost", username="root", password="8005", database="copppppp")

            my\_cursor = conn.cursor()

            my\_cursor.execute("insert into employee values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)", (

                self.var\_department.get(),

                self.var\_position.get(),

                self.var\_year.get(),

                self.var\_shift.get(),

                self.var\_employee\_id.get(),

                self.var\_employee\_name.get(),

                self.var\_gender.get(),

                self.var\_product\_category.get(),

                self.var\_phone.get(),

                self.var\_email.get(),

                self.var\_address.get(),

                self.var\_verification\_id.get()

            ))

            conn.commit()

            self.fetch\_data()

            conn.close()

            messagebox.showinfo("Success", "Data has been added successfully", parent=self.root)

        except Exception as e:

            messagebox.showerror("Error", f"Error due to: {str(e)}", parent=self.root)

        #Fetch Data from DATABASE

        # Fetch Data from DATABASE

    def fetch\_data(self):

            conn = mysql.connector.connect(host="localhost", username="root", password="8005", database="copppppp")

            my\_cursor = conn.cursor()

            my\_cursor.execute("SELECT Employee\_ID, Employee\_Name, Gender, Product\_Category, Phone, Email, Address, Verification\_ID FROM employee")

            data = my\_cursor.fetchall()

            if len(data) != 0:

               self.employee\_table.delete(\*self.employee\_table.get\_children())

               for row in data:

                   self.employee\_table.insert("", END, values=row)

            conn.commit()

            conn.close()

#Get cursor

    def get\_cursor(self,event=""):

        cursor\_focus=self.employee\_table.focus()

        content = self.employee\_table.item(cursor\_focus)

        data=content["values"]

        self.var\_department.set(data[0]),

        self.var\_position.set(data[1]),

        self.var\_year.set(data[2]),

        self.var\_shift.set(data[3]),

        self.var\_employee\_id.set(data[4]),

        self.var\_employee\_name.set(data[5]),

        self.var\_gender.set(data[6]),

        self.var\_product\_category.set(data[7]),

        self.var\_phone.set(data[8]),

        self.var\_email.set(data[9]),

        self.var\_address.set(data[10]),

        self.var\_verification\_id.set(data[11])

# update photo function

    def update\_data(self):

        if self.var\_department.get() == "Select Department" or self.var\_employee\_name.get() == "" or self.var\_employee\_id.get() == "":

         messagebox.showerror("Error", "All fields are required", parent=self.root)

        else:

            try:

                Update = messagebox.askyesno("Update","Do you want to update your employee details",parent =self.root)

                if Update>0:

                    conn = mysql.connector.connect(host="localhost", username="root", password="8005", database="copppppp")

                    my\_cursor = conn.cursor()

                    my\_cursor.execute("Upate employee set department =%s, position=%s,year=%s,shift=%s,employee\_name=%s,gender=%s,product\_category=%s,phone=%s,email=%s,address=%s,verification\_id=%s where employee\_id=%s",(

                                                            self.var\_department.get(),

                                                            self.var\_position.get(),

                                                            self.var\_year.get(),

                                                            self.var\_shift.get(),

                                                            self.var\_employee\_name.get(),

                                                            self.var\_gender.get(),

                                                            self.var\_product\_category.get(),

                                                            self.var\_phone.get(),

                                                            self.var\_email.get(),

                                                            self.var\_address.get(),

                                                            self.var\_verification\_id.get(),

                                                            self.var\_employee\_id.get()

                                                            ))

                else:

                    if not Update:

                        return

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

                conn.commit()

                self.fetch\_data()

                conn.close()

            except Exception as es:

                messagebox.showerror("Error",f"Due To:{str(es)}",parent=self.root)

#Delete function

    def delete\_data(self):

      if self.va\_std\_id.get()==" ":

        messagebox.showerror("Error","Employee id is required",parent=self.root)

      else:

        try:

            delete =messagebox.askyesno("Employee Delete Page","Do you want to delete this employee's details",parent=self.root)

            if delete>0:

                conn = mysql.connector.connect(host="localhost", username="root", password="8005", database="copppppp")

                my\_cursor = conn.cursor()

                sql="delete from student where employee\_id=%s"

                val =(self.var\_employee\_id.get(),)

                my\_cursor.execute(swl,val)

            else:

                if not  delete:

                    return

            conn.commit()

            self.fetch\_data()

            conn.close()

            messagebox.showinfo("Delete","Successfully Deleted Employee details",parent =self.root)

        except Exception as es:

                messagebox.showerror("Error",f"Due To:{str(es)}",parent=self.root)

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

    root = Tk()

    obj = Employee(root)

    root.mainloop()