*Name Samir Shukla*

*Sap id 500091591*

*CLOUD PERFORMANCE TUNING PROJECT*

*The title of my project is “****Attendance management System”***

*I have created this project in python language below is the code for the project*

import tkinter as tk

from tkinter import messagebox

import sqlite3 as sql

class AttendanceManager(tk.Tk):

    def \_\_init\_\_(self,\*args,\*\*kwargs):

        tk.Tk. \_\_init\_\_(self,\*args,\*\*kwargs)

        container=tk.Frame(self)

        container.pack(side="top",fill="both",expand=True)

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

        container.grid\_columnconfigure(0,weight=1)

        self.frames=dict()

        for F in (StartPage,NewRecord,ManageAttendance,DeleteRecord,EditRecord,AddSubjects,TodayData):

            frame=F(container,self)

            self.frames[F]=frame

            frame.grid(row=0,column=0,sticky="nsew")

        self.show\_frame(StartPage)

    def show\_frame(self,cont):

        frame=self.frames[cont]

        frame.tkraise()

class StartPage(tk.Frame):

    def \_\_init\_\_(self,parent,controller):

        tk.Frame.\_\_init\_\_(self,parent)

        label1=tk.Label(self,text="ATTENDANCE MANAGEMENT SYSTEM Project using Python",font=("Times",26))

        bt1=tk.Button(self,text="Add new record",font=("Times",16),height=2,width=17,bg="blue" ,command=lambda:controller.show\_frame(NewRecord))

        bt2=tk.Button(self,text="Manage attendance",font=("Times",16),height=2,width=17,bg="yellow",command=lambda:controller.show\_frame(ManageAttendance))

        bt3=tk.Button(self,text="Delete record",font=("Times",16),height=2,width=17,bg="red",command=lambda:controller.show\_frame(DeleteRecord))

        bt4=tk.Button(self,text="Edit record",font=("Times",16),height=2,width=17,bg="orange",command=lambda:controller.show\_frame(EditRecord))

        label1.pack()

        bt1.pack()

        bt2.pack()

        bt3.pack()

        bt4.pack()

class NewRecord(tk.Frame):

    def \_\_init\_\_(self,parent,controller):

        tk.Frame.\_\_init\_\_(self,parent)

        label1=tk.Label(self,text="New Record",font=("Times",24))

        label2=tk.Label(self,text="NOTE: If you want a new record, previous one will be deleted,continue?",font=("Times",14))

        bt2=tk.Button(self,text="YES",font=("Times",16),bg="orange",height=2,width=17,command=lambda:controller.show\_frame(AddSubjects))

        bt3=tk.Button(self,text="NO",font=("Times",16),bg="red",height=2,width=17,command=lambda:controller.show\_frame(StartPage))

        label1.pack()

        label2.pack()

        bt2.pack()

        bt3.pack()

class ManageAttendance(tk.Frame):

    def \_\_init\_\_(self,parent,controller):

        tk.Frame.\_\_init\_\_(self,parent)

        label1=tk.Label(self,text="Manage Attendance",font=("Times",24))

        label1.pack()

        bt2=tk.Button(self,text="Show status", bg="green",font=("Times",16),height=2,width=17,command=lambda:self.showstatus(controller))

        bt3=tk.Button(self,text="Today's data",  bg="orange",font=("Times",16),height=2,width=17,command=lambda:controller.show\_frame(TodayData))

        bt1=tk.Button(self,text="Back to home",bg="red",font=("Times",16),height=2,width=17,command=lambda:controller.show\_frame(StartPage))

        bt2.pack()

        bt3.pack()

        bt1.pack()

    def showstatus(self,controller):

        try:

            conn=sql.connect("attend")

            cur=conn.cursor()

            cur.execute('SELECT \* FROM attable')

            text=""

            for w in cur:

                if w[2]==0 and w[3]==0:

                    per="0"

                else:

                    per=w[2]/(w[2]+w[3])

                    per=per\*100

                    per=str(int(per))

                text=text+"sub id "+str(w[0])+" "+w[1]+" "+per+"%\n"

            messagebox.showinfo("status", text)

        except:

            messagebox.showinfo("Alert!", "There is no record")

class DeleteRecord(tk.Frame):

    def \_\_init\_\_(self,parent,controller):

        tk.Frame.\_\_init\_\_(self,parent)

        label1=tk.Label(self,text="Delete Record",font=("Times",24))

        label2=tk.Label(self,text="This action will delete the record,continue?",font=("Times",12))

        bt2=tk.Button(self,text="YES",bg="green",font=("Times",16),height=2,width=17,command=lambda:self.delrecord(controller))

        bt1=tk.Button(self,text="NO",bg="yellow",font=("Times",16),height=2,width=17,command=lambda:controller.show\_frame(StartPage))

        bt3=tk.Button(self,text="Back to home",bg="red",font=("Times",16),height=2,width=17,command=lambda:controller.show\_frame(StartPage))

        label1.pack()

        label2.pack()

        bt2.pack()

        bt1.pack()

        bt3.pack()

    def delrecord(self,controller):

        conn=sql.connect('attend')

        cur=conn.cursor()

        cur.execute('DROP TABLE IF EXISTS attable')

        conn.commit()

        conn.close()

        messagebox.showinfo("Alert!", "records deleted")

        controller.show\_frame(StartPage)

class EditRecord(tk.Frame):

    def \_\_init\_\_(self,parent,controller):

        tk.Frame.\_\_init\_\_(self,parent)

        label1=tk.Label(self,text="Edit Record",font=("Times",24))

        bt1=tk.Button(self,text="Back to home",bg="red",font=("Times",16),height=2,width=17,command=lambda:controller.show\_frame(StartPage))

        label1.pack()

        lb2=tk.Label(self,text="Input Subject ID: ",font=("Times",15))

        txt1=tk.Entry(self)

        lb2.pack()

        txt1.pack()

        lb3=tk.Label(self,text="Number of times attended:",font=("Times",15))

        txt2=tk.Entry(self)

        lb4=tk.Label(self,text="Number of times bunked:",font=("Times",15))

        txt3=tk.Entry(self)

        lb3.pack()

        txt2.pack()

        lb4.pack()

        txt3.pack()

        bt3=tk.Button(self,text="Update",bg="green",font=("Times",16),height=2,width=17,command=lambda:self.update(txt1.get(),txt2.get(),txt3.get()))

        bt2=tk.Button(self,text="Show subjects ID",bg="yellow",font=("Times",16),height=2,width=17,command=lambda:self.showid(controller))

        bt2.pack()

        bt3.pack()

        bt1.pack()

    def update(self,i,p,b):

        i=int(i)

        if p=="" or p==" " or p=="\n":

            p=0

        else:

            p=int(p)

        if b=="" or b==" " or b=="\n":

            b=0

        else:

            b=int(b)

        try:

            conn=sql.connect("attend")

            cur=conn.cursor()

            cur.execute("SELECT \* FROM attable WHERE subid=?",(i,))

            kk=cur.fetchone()

            np=p

            nb=b

            cur.execute("UPDATE attable SET attended = ? WHERE subid= ?",(np,i))

            cur.execute("UPDATE attable SET bunked = ? WHERE subid= ?",(nb,i))

            conn.commit()

            conn.close()

            messagebox.showinfo("Alert!", "Updated")

        except:

            messagebox.showinfo("Alert!", "There is no record")

    def showid(self,controller):

        try:

            conn=sql.connect("attend")

            cur=conn.cursor()

            cur.execute('SELECT \* FROM attable')

            text=""

            for w in cur:

                text=text+"sub id "+str(w[0])+" "+w[1]+"\n"

            messagebox.showinfo("Subject ID: ", text)

            conn.commit()

            conn.close()

        except:

            messagebox.showinfo("Alert!", "There is no record")

class AddSubjects(tk.Frame):

    def \_\_init\_\_(self,parent,controller):

        tk.Frame.\_\_init\_\_(self,parent)

        label1=tk.Label(self,text="Add subject's name seperated by commas(,)",font=("Times",16))

        txt1=tk.Text(self,font=("Times",16),width=48,height=3)

        bt2=tk.Button(self,text="Add subjects!",bg="orange",font=("Times",16),height=2,width=17,command=lambda:self.addsub(txt1.get("1.0",tk.END),controller))

        bt1=tk.Button(self,text="Back to home",bg="red",font=("Times",16),height=2,width=17,command=lambda:controller.show\_frame(StartPage))

        label1.pack()

        txt1.pack()

        bt2.pack()

        bt1.pack()

    def addsub(self,a,controller):

        conn=sql.connect('attend')

        cur=conn.cursor()

        cur.execute('DROP TABLE IF EXISTS attable')

        a=a[0:len(a)-1]

        a=a.split(",")

        if len(a)==1 and a[0]=="" :

            messagebox.showinfo("Alert!", "Please enter the subjects")

        else:

            sid=1

            cur.execute('CREATE TABLE attable(subid INTEGER,subject TEXT,attended INTEGER,bunked INTEGER)')

            for sub in a:

                cur.execute('INSERT INTO attable (subid,subject,attended,bunked) VALUES(?,?,?,?)',(sid,sub,0,0))

                sid=sid+1

            conn.commit()

            conn.close()

            messagebox.showinfo("congratulations!", "subjects are added")

            controller.show\_frame(StartPage)

class TodayData(tk.Frame):

    def \_\_init\_\_(self,parent,controller):

        tk.Frame.\_\_init\_\_(self,parent)

        label1=tk.Label(self,text="Enter data of today",font=("Times",24))

        label1.pack()

        bt2=tk.Button(self,text="Show id of subjects",bg="yellow",font=("Times",16),height=2,width=17,command=lambda:self.showid(controller))

        bt1=tk.Button(self,text="Back to home",bg="red",font=("Times",16),height=2,width=17,command=lambda:controller.show\_frame(StartPage))

        lb2=tk.Label(self,text="Input the corresponding Subject ID: ",font=("Times",15))

        txt1=tk.Entry(self)

        lb2.pack()

        txt1.pack()

        lb3=tk.Label(self,text="Number of times attended:",font=("Times",15))

        txt2=tk.Entry(self)

        lb4=tk.Label(self,text="Number of times bunked:",font=("Times",15))

        txt3=tk.Entry(self)

        lb3.pack()

        txt2.pack()

        lb4.pack()

        txt3.pack()

        bt3=tk.Button(self,text="Add to record",bg="orange",font=("Times",16),height=2,width=17,command=lambda:self.addrecord(txt1.get(),txt2.get(),txt3.get()))

        bt3.pack()

        bt2.pack()

        bt1.pack()

    def showid(self,controller):

        try:

            conn=sql.connect("attend")

            cur=conn.cursor()

            cur.execute('SELECT \* FROM attable')

            text=""

            for w in cur:

                text=text+"sub id "+str(w[0])+" "+w[1]+"\n"

            messagebox.showinfo("Subject ID: ", text)

            conn.commit()

            conn.close()

        except:

            messagebox.showinfo("Alert!", "There is no record")

    def addrecord(self,i,p,b):

        i=int(i)

        if p=="" or p==" " or p=="\n":

            p=0

        else:

            p=int(p)

        if b=="" or b==" " or b=="\n":

            b=0

        else:

            b=int(b)

        try:

            conn=sql.connect("attend")

            cur=conn.cursor()

            cur.execute("SELECT \* FROM attable WHERE subid=?",(i,))

            kk=cur.fetchone()

            np=kk[2]+p

            nb=kk[3]+b

            cur.execute("UPDATE attable SET attended = ? WHERE subid= ?",(np,i))

            cur.execute("UPDATE attable SET bunked = ? WHERE subid= ?",(nb,i))

            conn.commit()

            conn.close()

            messagebox.showinfo("Alert!", "Done")

        except:

            messagebox.showinfo("Alert!", "There is no record")

def main():

    app=AttendanceManager()

    app.title("Attendance Management - CopyAssignment")

    app.mainloop()

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

    main()

***here is the explanation of the code***

This code is a simple Attendance Management System implemented using the Tkinter library in Python. It consists of a GUI application with various frames for different functionalities.

**Imports:**

tkinter: GUI library for creating the graphical user interface.

messagebox: Part of the tkinter library, used to display message boxes.

sqlite3: SQLite database library for handling the database operations.

**Class Definitions:**

AttendanceManager: Inherits from tk.Tk and serves as the main application class. It manages different frames for various functionalities.

StartPage, NewRecord, ManageAttendance, DeleteRecord, EditRecord, AddSubjects, TodayData: Different frames for different sections of the application.

**Initialization:**

The AttendanceManager class initializes with a set of buttons for different functionalities (Add new record, Manage attendance, Delete record, Edit record) on the start page.

**Frame Switching:**

The show\_frame method in the AttendanceManager class is used to switch between frames. It takes a frame class as an argument and raises that frame to the top.

**Functionality Descriptions:**

NewRecord: Asks the user if they want to create a new record, and if yes, it redirects to the AddSubjects frame to add subjects.

ManageAttendance: Provides options to show the attendance status and today's data. Uses SQLite to fetch and display data.

DeleteRecord: Asks the user if they want to delete the record. If yes, it deletes the entire table from the database.

EditRecord: Allows the user to update the attendance data for a specific subject.

AddSubjects: Allows the user to input subject names separated by commas and adds them to the database.

TodayData: Allows the user to input attendance data for a specific subject for the current day.

**Database Operations:**

The code uses SQLite to create and manage a database named 'attend' and a table named 'attable' to store subject information and attendance data.

**Main Function:**

The main function creates an instance of the AttendanceManager class, sets the window title, and starts the main event loop.

Than I have deployed the project on cloud platform:

