Source Code Of RMS Project

Course.py

```
from tkinter import*
from PIL import Image,ImageTk  #pip install pillow
from tkinter import ttk,messagebox
import sqlite3
class CourseClass:
    def __init__(self,root):
        self.root=root
        self.root.title("Student Result Management System")
        self.root.geometry("1200x480+80+170")
        self.root.config(bg="white")
        self.root.focus_force()
        #=====title======
        title=Label(self.root,text="Manage Course Details",font=("goudy old style",20,"bold"),bg="#033054",fg="white").place(x=10,y=15,width=1180,height=35)
        #=======Variables======
        self.var_course=StringVar()
        self.var_duration=StringVar()
        self.var_charges=StringVar()



        #========Widgets ======================
        lbl_courseName=Label(self.root,text="Course Name", font=("goudy old style",15,'bold'),bg='white').place(x=10,y=60)
        lbl_duration=Label(self.root,text="Duration", font=("goudy old style",15,'bold'),bg='white').place(x=10,y=100)
        lbl_charges=Label(self.root,text="Charges", font=("goudy old style",15,'bold'),bg='white').place(x=10,y=140)
```

```
        lbl_description=Label(self.root,text="Description", font=("goudy old
style",15,'bold'),bg='white').place(x=10,y=180)


        #========Entry Fields=======


        self.txt_courseName=Entry(self.root,textvariable=self.var_course, font=("goudy
old style",15,'bold'),bg='lightyellow')

        self.txt_courseName.place(x=150,y=60,width=200)

        txt_duration=Entry(self.root,textvariable=self.var_duration,font=("goudy old
style",15,'bold'),bg='lightyellow').place(x=150,y=100,width=200)

        txt_charges=Entry(self.root,textvariable=self.var_charges, font=("goudy old
style",15,'bold'),bg='lightyellow').place(x=150,y=140,width=200)

        self.txt_description=Text(self.root, font=("goudy old
style",15,'bold'),bg='lightyellow')

        self.txt_description.place(x=150,y=180,width=500,height=130)


        #=====Buttons======


        self.btn_add=Button(self.root,text='Save',font=("goudy old
style",15,"bold"),bg="#2196f3",fg="white",cursor="hand2",command=self.add)

        self.btn_add.place(x=150,y=400,width=110,height=40)

        self.btn_update=Button(self.root,text='Update',font=("goudy old
style",15,"bold"),bg="#4caf50",fg="white",cursor="hand2",command=self.update)

        self.btn_update.place(x=270,y=400,width=110,height=40)

        self.btn_delete=Button(self.root,text='Delete',font=("goudy old
style",15,"bold"),bg="#f44336",fg="white",cursor="hand2",command=self.delete)

        self.btn_delete.place(x=390,y=400,width=110,height=40)

        self.btn_clear=Button(self.root,text='Clear',font=("goudy old
style",15,"bold"),bg="#607d8b",fg="white",cursor="hand2",command=self.clear)

        self.btn_clear.place(x=510,y=400,width=110,height=40)

        #=======Search Panel==========

        self.var_search=StringVar()
```

```
        lbl_search_courseName=Label(self.root,text=" Course Name ", font=("goudy old style",15,'bold'),bg='white').place(x=720,y=60)

        txt_search_courseName=Entry(self.root,textvariable=self.var_search, font=("goudy old style",15,'bold'),bg='lightyellow').place(x=870,y=60,width=180)

        btn_search=Button(self.root,text='Search',font=("goudy old style",15,"bold"),bg="#03a9f4",fg="white",cursor="hand2",command=self.search).place(x=1070,y=60,width=120,height=28)


        #========content============

        self.C_Frame=Frame(self.root,bd=2,relief=RIDGE)

        self.C_Frame.place(x=720,y=100,width=470,height=340)


        scrolly=Scrollbar(self.C_Frame,orient=VERTICAL)

        scrollx=Scrollbar(self.C_Frame,orient=HORIZONTAL)

        self.CourseTable=ttk.Treeview(self.C_Frame,columns=("cid","name","duration","charges","description"),xscrollcommand=scrollx.set,yscrollcommand=scrolly.set)


        scrollx.pack(side=BOTTOM,fill=X)

        scrolly.pack(side=RIGHT,fill=Y)

        scrollx.config(command=self.CourseTable.xview)

        scrolly.config(command=self.CourseTable.yview)



        self.CourseTable.heading("cid",text="Course ID")

        self.CourseTable.heading("name",text="Name")

        self.CourseTable.heading("duration",text=" Duration")

        self.CourseTable.heading("charges",text=" Charges")

        self.CourseTable.heading("description",text="Description")

        self.CourseTable["show"]='headings'

        self.CourseTable.column("cid",width=100)

        self.CourseTable.column("name",width=100)

        self.CourseTable.column("duration",width=100)
```

```
        self.CourseTable.column("charges",width=100)
        self.CourseTable.column("description",width=150)
        self.CourseTable.pack(fill=BOTH,expand=1)
        self.CourseTable.bind("<ButtonRelease-1>",self.get_data)
        self.show()

#================================================================================


    def clear(self):
        self.show()
        self.var_course.set("")
        self.var_duration.set("")
        self.var_charges.set("")
        self.var_search.set("")
        self.txt_description.delete('1.0',END)
        self.txt_courseName.config(state=NORMAL)
    def delete(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            if self.var_course.get()=="":
                messagebox.showerror("Error","Course Name should be
required",parent=self.root)
            else:
                cur.execute("select * from course where name=?",(self.var_course.get(),))
                row=cur.fetchone()
                print(row)
                if row==None:
                    messagebox.showerror("Error","please select course from list
",parent=self.root)
```

```
            else:
                op=messagebox.askyesno("Confirm","Do you really want to
delete?",parent=self.root)
                if op==True:
                    cur.execute("delete from course where
name=?",(self.var_course.get(),))
                    con.commit()
                    messagebox.showinfo("Delete","Course deleted
Successfully",parent=self.root)
                    self.clear()

        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")



    def get_data(self,ev):
        self.txt_courseName.config(state='readonly')
        self.txt_courseName
        r=self.CourseTable.focus()
        content=self.CourseTable.item(r)
        row=content["values"]
        #print(row)
        self.var_course.set(row[1])
        self.var_duration.set(row[2])
        self.var_charges.set(row[3])
        #self.var_course.set(row[4])
        self.txt_description.delete('1.0',END)
        self.txt_description.insert(END,row[4])


    def add(self):
```

```
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            if self.var_course.get()=="":
                messagebox.showerror("Error","Course Name should be
required",parent=self.root)
            else:
                cur.execute("select * from course where name=?",(self.var_course.get(),))
                row=cur.fetchone()
                print(row)
                if row!=None:
                    messagebox.showerror("Error","Course Name already
avilable",parent=self.root)
                else:
                    cur.execute("insert into course(name,duration,charges,description)
values(?,?,?,?)",(
                        self.var_course.get(),
                        self.var_duration.get(),
                        self.var_charges.get(),
                        self.txt_description.get("1.0",END)
                    ))
                    con.commit()
                    messagebox.showinfo("Success","Course Added
Successfully",parent=self.root)
                    self.show()
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")


    def update(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
```

```
        try:
            if self.var_course.get()=="":
                messagebox.showerror("Error","Course Name should be
required",parent=self.root)
            else:
                cur.execute("select * from course where name=?",(self.var_course.get(),))
                row=cur.fetchone()
                print(row)
                if row==None:
                    messagebox.showerror("Error","Select Course from
list",parent=self.root)
                else:
                    cur.execute("update course set
duration=?,charges=?,description=?where name=?",(
                        self.var_duration.get(),
                        self.var_charges.get(),
                        self.txt_description.get("1.0",END),
                        self.var_course.get()
                    ))
                    con.commit()
                    messagebox.showinfo("Success","Course update
Successfully",parent=self.root)
                    self.show()
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")


    def show(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            cur.execute("select * from course ")
```

```
            rows=cur.fetchall()
            self.CourseTable.delete(*self.CourseTable.get_children())
            for row in rows:
                self.CourseTable.insert('',END,values=row)

        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")

    def search(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            cur.execute(f"select * from course where name LIKE
'%{self.var_search.get()}%'")
            rows=cur.fetchall()
            self.CourseTable.delete(*self.CourseTable.get_children())
            for row in rows:
                self.CourseTable.insert('',END,values=row)

        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")



if __name__=="__main__":
    root=Tk()
    obj=CourseClass(root)
    root.mainloop()
```