Source Code Of RMS Project

Login.py

```
from tkinter import*
from tkinter import ttk
from PIL import Image,ImageTk
from tkinter import messagebox
import random
import time
import datetime
import mysql.connector
import sqlite3


def main():
    win=Tk()
    app=Login_window(win)
    win.mainloop()



class Login_window:
    def __init__(self,root):
        self.root=root
        self.root.title("Login")
        self.root.geometry("1530x800+0+0")


        self.var_email=StringVar()


        self.bg=ImageTk.PhotoImage(file="images/image1.png")


        lbl_bg=Label(self.root,image=self.bg)
```

```
        lbl_bg.place(x=0,y=0,relwidth=1,relheight=1)


        frame=Frame(self.root,bg="black")
        frame.place(x=610,y=170,width=340,height=500)



        get_str=Label(frame,text="Login-Form",font=("times new
roman",20,"bold"),fg="white",bg="black")
        get_str.place(x=95,y=50)


        #Entry


        username=lbl=Label(frame,text="UserName",font=("times new
roman",20,"bold"),fg="white",bg="black")
        username.place(x=40,y=150)
        self.txtuser=ttk.Entry(frame,font=("times new roman",15,"bold"))
        self.txtuser.place(x=40,y=200,width=270)


        password=lbl=Label(frame,text="Password",font=("times new
roman",20,"bold"),fg="white",bg="black")
        password.place(x=40,y=245)
        self.txtpass=ttk.Entry(frame,font=("times new roman",15,"bold"))
        self.txtpass.place(x=40,y=290,width=270)
#==========Login-Button=========================================
        loginbtn=Button(frame,command=self.login,text="Login",font=("times new
roman",15,"bold"),bd=3,relief=RIDGE,fg="white",bg="red")
        loginbtn.place(x=110,y=340,width=120,height=35)


        #============register -Button==================
        registerbtn=Button(frame,text="New User
Register",command=self.register_window,font=("times new
roman",10,"bold"),borderwidth=0,fg="white",bg="black")
```

```
        registerbtn.place(x=15,y=390,width=160)


        #=====Forgot-Bitton=============
        Forgotbtn=Button(frame,text="Forgot
Password",command=self.forgot_password_window,font=("times new
roman",10,"bold"),borderwidth=0,fg="white",bg="black")
        Forgotbtn.place(x=10,y=420,width=160)


    def register_window(self):
        self.new_window=Toplevel(self.root)
        self.app=Register(self.new_window)


    def login(self):
        if self.txtuser.get()=="" or self.txtpass.get()=="":
            messagebox.showerror("Error","All fields Are Required")
        elif self.txtuser.get()=="ansh" and self.txtpass.get()=="123":


            messagebox.showinfo("Success", "Wlecome to my world")
        else:
            conn=mysql.connector.connect(host="localhost",user="root",password="Ansh
@9026",database="my_data")
            my_cursor=conn.cursor()
            my_cursor.execute("select * from register where email=%s and
password=%s",(
                                                    self.txtuser.get(),
                                                    self.txtpass.get()
                                                   ))


            row=my_cursor.fetchone()
            if row==None:
                messagebox.showerror("Error","Invaild Username And Password")
```

```
            else:
                open_main=messagebox.askyesno("YesNo","Access Only Admin")
                if open_main>0:
                    self.new_window=Toplevel(self.root)
                    self.app=RMS(self.new_window)
                else:
                    if not open_main:
                        return
            conn.commit()
            conn.close()
            #=======================reset pass=====================
    def reset_pass(self):
        if self.combo_security_Q.get()=="Select":
            messagebox.showerror('Error',"Select Security Question",parent=self.root2)
        elif self.txt_security.get()=="":
            messagebox.showerror("Error","Please enter the answer",parent=self.root2)
        elif self.txt_new_password.get()=="":
            messagebox.showerror("Error","Please Enter the New
PAssword",parent=self.root2)
        else:
            conn=mysql.connector.connect(host="localhost",user="root",password="Ansh
@9026",database="my_data")
            my_cursor=conn.cursor()
            qury=("select * from register where email=%s and securityQ=%s and
securityA=%s")
            value=(self.txtuser.get(),self.combo_security_Q.get(),self.txt_security.get(),)
            my_cursor.execute(qury,value)
            row=my_cursor.fetchone()
            if row==None:
                messagebox.showerror("Error","Please Enter the Correct
Answer",parent=self.root2)
```

```
            else:
                query=("update register set password=%s where email=%s")
                value=(self.txt_new_password.get(),self.txtuser.get())
                my_cursor.execute(query,value)


                conn.commit()
                conn.close()
                messagebox.showinfo("Info","Your password has been reset, please login
new password",parent=self.root2)
                self.root2.destroy()




            #====forgot password window=======================


    def forgot_password_window(self):
        if self.txtuser.get()=="":
            messagebox.showerror("Error","Please Enter The Email Address To Reset
Password")
        else:
            conn=mysql.connector.connect(host="localhost",user="root",password="Ansh
@9026",database="my_data")
            my_cursor=conn.cursor()
            query=("select * from register where email=%s")
            value=(self.txtuser.get(),)
            my_cursor.execute(query,value)
            row=my_cursor.fetchone()
            #print(row)


            if row == None:
                messagebox.showerror("My Error","Please Enter the Vaild Username")
```

```
        else:
            conn.close()
            self.root2=Toplevel()
            self.root2.title("Forgot Password")
            self.root2.geometry("340x450+610+170")

            l=Label(self.root2,text="Forgot Password",font=("times new
roman",15,"bold"),fg="red",bg="white")
            l.place(x=0,y=10,relwidth=1)

            security_Q=Label(self.root2,text="Security Question",font=("times new
roman",15,"bold"),bg="white")
            security_Q.place(x=50,y=80)

            self.combo_security_Q=ttk.Combobox(self.root2,font=("time new
roman",15,"bold"),state="readonly")
            self.combo_security_Q["values"]=("Select","Your Birth Place", "Your
Girlfriend Name", "Your Pet Name")
            self.combo_security_Q.place(x=50,y=110,width=250)
            self.combo_security_Q.current(0)

            security_A=Label(self.root2,text="Security Answer",font=("times new
roman",15,"bold"),bg="white")
            security_A.place(x=50,y=150)

            self.txt_security=ttk.Entry(self.root2,font=("times new roman",15,"bold"))
            self.txt_security.place(x=50,y=180,width=250)

            new_password=Label(self.root2,text="New Password",font=("times new
roman",15,"bold"),bg="white")
            new_password.place(x=50,y=220)
```

```
        self.txt_new_password=ttk.Entry(self.root2,font=("times new
roman",15,"bold"))
        self.txt_new_password.place(x=50,y=250,width=250)


        btn=Button(self.root2,text="Reset",command=self.reset_pass,font=("times
new roman",15,"bold"),fg="white",bg="green")
        btn.place(x=150,y=290)









class Register:
    def __init__(self,root):
        self.root=root
        self.root.title("Register")
        self.root.geometry("1530x800+0+0")


        #=========variables============================
        self.var_fname=StringVar()
        self.var_lname=StringVar()
        self.var_contact=StringVar()
        self.var_email=StringVar()
        self.var_securityQ=StringVar()
        self.var_SecurityA=StringVar()
```

```
        self.var_pass=StringVar()
        self.var_confpass=StringVar()




        #====================background image=======================
        self.bg=ImageTk.PhotoImage(file="images/image1.png")
        lbl_bg=Label(self.root,image=self.bg)
        lbl_bg.place(x=0,y=0,relwidth=1,relheight=1)
        #================= left image================================
        self.bg1=ImageTk.PhotoImage(file="images/side.png")
        left_lbl=Label(self.root,image=self.bg1)
        left_lbl.place(x=150,y=100,width=410,height=500)
        frame=Frame(self.root,bg="white")
        frame.place(x=560,y=100,width=800,height=500)
        #==============Main Frame=======================
        register_lbl=Label(frame,text="REGISTER HERE",font=("times new
roman",20,"bold"),fg="green",bg="white")
        register_lbl.place(x=20,y=20)
        #=======Entry field============================


        fname=Label(frame,text="First Name",font=("times new
roman",15,"bold"),bg="white")
        fname.place(x=50,y=100)


        self.fname_entry=ttk.Entry(frame,textvariable=self.var_fname,font=("times new
roman",15))
        self.fname_entry.place(x=50,y=130,width=250)
```

```
        l_name=Label(frame,text="Last Name",font=("times new
roman",15,"bold"),bg="white")

        l_name.place(x=370,y=100)


        self.txt_lname=ttk.Entry(frame,textvariable=self.var_lname,font=("times new
roman",15,))

        self.txt_lname.place(x=370,y=130,width=250)


        contact=Label(frame,text="Contact No",font=("times new
roman",15,"bold"),bg="white")

        contact.place(x=50,y=170)


        self.txt_contact=ttk.Entry(frame,textvariable=self.var_contact,font=("times new
roman",15,"bold"))

        self.txt_contact.place(x=50,y=200,width=250)


        email=Label(frame,text="Email",font=("times new roman",15,"bold"),bg="white")

        email.place(x=370,y=170)


        self.txt_email=ttk.Entry(frame,textvariable=self.var_email,font=("times new
roman",15,"bold"))

        self.txt_email.place(x=370,y=200,width=250)


        security_Q=Label(frame,text="Security Question",font=("times new
roman",15,"bold"),bg="white")

        security_Q.place(x=50,y=240)


        self.combo_security_Q=ttk.Combobox(frame,textvariable=self.var_securityQ,font
=("time new roman",15,"bold"),state="readonly")

        self.combo_security_Q["values"]=("Select","Your Birth Place", "Your Girlfriend
Name", "Your Pet Name")

        self.combo_security_Q.place(x=50,y=270,width=250)
```

```
        self.combo_security_Q.current(0)



        security_A=Label(frame,text="Security Answer",font=("times new
roman",15,"bold"),bg="white")

        security_A.place(x=370,y=240)


        self.txt_security=ttk.Entry(frame,textvariable=self.var_SecurityA,font=("times
new roman",15,"bold"))

        self.txt_security.place(x=370,y=270,width=250)


        pswd=Label(frame,text="Password",font=("times new
roman",15,"bold"),bg="white")

        pswd.place(x=50,y=310)


        self.txt_pswd=ttk.Entry(frame,textvariable=self.var_pass,font=("times new
roman",15,"bold"))

        self.txt_pswd.place(x=50,y=340,width=250)


        confirm_pswd=Label(frame,text="Confirm Password",font=("times new
roman",15,"bold"),bg="white")

        confirm_pswd.place(x=370,y=310)


        self.txt_confirm_pswd=ttk.Entry(frame,textvariable=self.var_confpass,font=("tim
es new roman",15,"bold"))

        self.txt_confirm_pswd.place(x=370,y=340,width=250)


        #=============Check Button======================

        self.var_check=IntVar()

        checkbtn=Checkbutton(frame,variable=self.var_check,text="I Agree The Terms &
Condition",font=("times new roman",12,"bold"),onvalue=1,offvalue=0)

        checkbtn.place(x=50,y=380)
```

```
        #===========button=================

        img=Image.open("images/register.png")

        img=img.resize((200,50),Image.ANTIALIAS)

        self.photoimage=ImageTk.PhotoImage(img)

        b1=Button(frame,image=self.photoimage,command=self.register_data,borderwid
th=0,cursor="hand2",font=("times new roman",12,"bold"))

        b1.place(x=50,y=420,width=200)


        img1=Image.open("images/login.png")

        img1=img1.resize((200,50),Image.ANTIALIAS)

        self.photoimage1=ImageTk.PhotoImage(img1)

        b1=Button(frame,image=self.photoimage1,command=self.return_login,borderwid
th=0,cursor="hand2",font=("times new roman",12,"bold"))

        b1.place(x=370,y=420,width=200)


        #====================function================================

    def register_data(self):

        if self.var_fname.get()==""or self.var_email.get()==""or
self.var_securityQ.get()=="Select":

            messagebox.showerror("Error","All fields are required")

        elif self.var_pass.get()!=self.var_confpass.get():

            messagebox.showerror("Error","password & confirm password must be
same ")

        elif self.var_check.get()==0:

            messagebox.showerror("Error","Please agree our terms and conditions")

        else:

            conn=mysql.connector.connect(host="localhost",user="root",password="Ansh
@9026",database="my_data")

            my_cursor=conn.cursor()

            query=("select * from register where email=%s")
```

```
            value=(self.var_email.get(),)
            my_cursor.execute(query,value)
            row=my_cursor.fetchone()
            if row!=None:
                messagebox.showerror("Error","User Already Exist, Please try another
email")
            else:
                my_cursor.execute("insert into register values(%s,%s,%s,%s,%s,%s,%s)",(
                                                        self.var_fname.get(),
                                                        self.var_lname.get(),
                                                        self.var_contact.get(),
                                                        self.var_email.get(),
                                                        self.var_securityQ.get(),
                                                        self.var_SecurityA.get(),
                                                        self.var_pass.get()


                                                        ))
            conn.commit()
            conn.close()
            messagebox.showinfo("Success","Register Successfully")

    def return_login(self):
        self.root.destroy()


class RMS:

    def __init__(self,root):
        self.root=root
```

```
        self.root.title("Student Result Management System")
        self.root.geometry("1550x700+0+0")
        self.root.config(bg="white")

        #=====icons=====
        self.logo_dash=ImageTk.PhotoImage(file="images/logo_p.png")

       #=====title======
        title=Label(self.root,text="Student Result Management System",padx=10,compound=LEFT,image=self.logo_dash,font=("goudy old style",20,"bold"),bg="#033054",fg="white").place(x=0,y=0,relwidth=1,height=50)
        #========Menu=====
        M_Frame=LabelFrame(self.root,text="Menus",font=("times new roman",15),bg="white")
        M_Frame.place(x=10,y=70,width=1540,height=80)

        btn_course=Button(M_Frame,text="Course",font=("goudy old style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add_course).place(x=20,y=5,width=200,height=40)
        btn_student=Button(M_Frame,text="Student",font=("goudy old style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add_student).place(x=440,y=5,width=200,height=40)
        btn_result=Button(M_Frame,text="Result",font=("goudy old style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add_result).place(x=900,y=5,width=200,height=40)
        btn_view=Button(M_Frame,text="ViewStudent Result",font=("goudy old style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add_report).place(x=1280,y=5,width=200,height=40)

        #=====Content window=========
        self.bg_img=Image.open("images/bg.png")
        self.bg_img=self.bg_img.resize((1520,350),Image.ANTIALIAS)
        self.bg_img=ImageTk.PhotoImage(self.bg_img)
```

```
        self.lbl_bg=Label(self.root,image=self.bg_img).place(x=320,y=200,width=920,heig
ht=350)









        #=====footer======

        footer=Label(self.root,text="SRMS-Student Result Management System\n
Contact Us for any Issue:  9026049848\n @Ansh | All Rights Reserved",font=("goudy
old style",12,),bg="#262626",fg="white").pack(side=BOTTOM,fill=X)









    def add_course(self):

        self.new_win=Toplevel(self.root)

        self.new_obj=CourseClass(self.new_win)


    def add_student(self):

        self.new_win=Toplevel(self.root)

        self.new_obj=studentClass(self.new_win)


    def add_result(self):

        self.new_win=Toplevel(self.root)

        self.new_obj=resultClass(self.new_win)


    def add_report(self):

        self.new_win=Toplevel(self.root)

        self.new_obj=reportClass(self.new_win)
```

```
class resultClass:
    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="Add Student Result",font=("goudy old
style",20,"bold"),bg="orange",fg="#262626").place(x=10,y=15,width=1180,height=50)
        #======widgets==========
        #========variables===============
        self.var_roll=StringVar()
        self.var_name=StringVar()
        self.var_course=StringVar()
        self.var_marks=StringVar()
        self.var_full_marks=StringVar()
        self.roll_list=[]
        self.fetch_roll()


        lbl_select=Label(self.root,text="Select Student", font=("goudy old
style",20,"bold"),bg="white").place(x=50,y=100)
        lbl_name=Label(self.root,text="Name", font=("goudy old
style",20,"bold"),bg="white").place(x=50,y=160)
        lbl_course=Label(self.root,text="Course", font=("goudy old
style",20,"bold"),bg="white").place(x=50,y=220)
        lbl_marks_ob=Label(self.root,text="Marks Obtained", font=("goudy old
style",20,"bold"),bg="white").place(x=50,y=280)
        lbl_full_marks=Label(self.root,text="Full Marks", font=("goudy old
style",20,"bold"),bg="white").place(x=50,y=340)
```

```
        self.txt_student=ttk.Combobox(self.root,textvariable=self.var_roll,values=self.rol
l_list, font=("goudy old style",15,'bold'),state='readonly',justify=CENTER)

        self.txt_student.place(x=280,y=100,width=200)

        self.txt_student.set("Select")

        btn_search=Button(self.root,text='Search',font=("goudy old
style",15,"bold"),bg="#03a9f4",fg="white",cursor="hand2",command=self.search).plac
e(x=500,y=100,width=100,height=28)


        txt_name=Entry(self.root,textvariable=self.var_name, font=("goudy old
style",20,'bold'),bg='lightyellow',state='readonly').place(x=280,y=160,width=320)

        txt_course=Entry(self.root,textvariable=self.var_course, font=("goudy old
style",20,'bold'),bg='lightyellow',state='readonly').place(x=280,y=220,width=320)

        txt_marks=Entry(self.root,textvariable=self.var_marks, font=("goudy old
style",20,'bold'),bg='lightyellow').place(x=280,y=280,width=320)

        txt_full_marks=Entry(self.root,textvariable=self.var_full_marks, font=("goudy old
style",20,'bold'),bg='lightyellow').place(x=280,y=340,width=320)


        #===========button============

        btn_add=Button(self.root,text="Submit",font=("times new
roman",15),bg="lightgreen",activebackground="lightgreen",cursor="hand2",command
=self.add).place(x=300,y=420,width=120,height=35)

        btn_clear=Button(self.root,text="Clear",font=("times new
roman",15),bg="lightgray",activebackground="lightgray",cursor="hand2",command=s
elf.clear).place(x=430,y=420,width=120,height=35)


        #==========image==============

        self.bg_img=Image.open("images/result.jpg")

        self.bg_img=self.bg_img.resize((500,350),Image.ANTIALIAS)

        self.bg_img=ImageTk.PhotoImage(self.bg_img)


        self.lbl_bg=Label(self.root,image=self.bg_img).place(x=650,y=100)


    #============================================================================
```

```
    def fetch_roll(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            cur.execute("select roll from student")
            rows=cur.fetchall()
            if len(rows)>0:
                for row in rows:
                    self.roll_list.append(row[0])
        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 name,course from student where
roll=?",(self.var_roll.get(),))
            row=cur.fetchone()
            if row!=None:
                self.var_name.set(row[0])
                self.var_course.set(row[1])
            else:
                messagebox.showerror("Error","No record found",parent=self.root)



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


    def add(self):
```

```
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            if self.var_name.get()=="":
                messagebox.showerror("Error","please first search student
record",parent=self.root)
            else:
                cur.execute("select * from result where roll=? and
course=?",(self.var_roll.get(),self.var_course.get(),))
                row=cur.fetchone()
                print(row)
                if row!=None:
                    messagebox.showerror("Error","Result already
avilable",parent=self.root)
                else:
                    per=(int(self.var_marks.get())*100)/int(self.var_full_marks.get())
                    cur.execute("insert into
result(roll,name,course,marks_ob,full_marks,per) values(?,?,?,?,?,?)",(
                        self.var_roll.get(),
                        self.var_name.get(),
                        self.var_course.get(),
                        self.var_marks.get(),
                        self.var_full_marks.get(),
                        str(per)
                    ))
                    con.commit()
                    messagebox.showinfo("Success","Result Added
Successfully",parent=self.root)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")
```

```
    def clear(self):
        self.var_roll.set("Select"),
        self.var_name.set(""),
        self.var_course.set(""),
        self.var_marks.set(""),
        self.var_full_marks.set("")


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)}")

class reportClass:
    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="View Student Result",font=("goudy old
style",20,"bold"),bg="orange",fg="#262626").place(x=10,y=15,width=1180,height=50)

        #============search====================

        self.var_search=StringVar()

        self.var_id=""

        lbl_search=Label(self.root,text="Search by roll no.", font=("goudy old
style",20,"bold"),bg="white").place(x=280,y=100)

        txt_search=Entry(self.root,textvariable=self.var_search, font=("goudy old
style",20),bg="lightyellow").place(x=520,y=100,width=150)

        btn_search=Button(self.root,text='Search',font=("goudy old
style",15,"bold"),bg="#03a9f4",fg="white",cursor="hand2",command=self.search).plac
e(x=680,y=100,width=100,height=35)

        btn_clear=Button(self.root,text='clear',font=("goudy old
style",15,"bold"),bg="gray",fg="white",cursor="hand2",command=self.clear).place(x=80
0,y=100,width=100,height=35)


        #=========results_label==============

        lbl_roll=Label(self.root,text="Roll No",font=("goudy old
style",15,"bold"),bg="white",bd=2,relief=GROOVE).place(x=150,y=230,width=150,height=5
0)

        lbl_name=Label(self.root,text="Name",font=("goudy old
style",15,'bold'),bg="white",bd=2,relief=GROOVE).place(x=300,y=230,width=150,height=5
0)

        lbl_course=Label(self.root,text="Course",font=("goudy old
style",15,'bold'),bg="white",bd=2,relief=GROOVE).place(x=450,y=230,width=150,height=5
0)

        lbl_marks=Label(self.root,text="Marks Obtained",font=("goudy old
style",15,'bold'),bg="white",bd=2,relief=GROOVE).place(x=600,y=230,width=150,height=5
0)

        lbl_full=Label(self.root,text="Total Marks",font=("goudy old
style",15,'bold'),bg="white",bd=2,relief=GROOVE).place(x=750,y=230,width=150,height=5
0)

        lbl_per=Label(self.root,text="Percentage",font=("goudy old
style",15,'bold'),bg="white",bd=2,relief=GROOVE).place(x=900,y=230,width=150,height=5
0)
```

```
        self.roll=Label(self.root,font=("goudy old
style",15,"bold"),bg="white",bd=2,relief=GROOVE)

        self.roll.place(x=150,y=280,width=150,height=50)

        self.name=Label(self.root,font=("goudy old
style",15,'bold'),bg="white",bd=2,relief=GROOVE)

        self.name.place(x=300,y=280,width=150,height=50)

        self.course=Label(self.root,font=("goudy old
style",15,'bold'),bg="white",bd=2,relief=GROOVE)

        self.course.place(x=450,y=280,width=150,height=50)

        self.marks=Label(self.root,font=("goudy old
style",15,'bold'),bg="white",bd=2,relief=GROOVE)

        self.marks.place(x=600,y=280,width=150,height=50)

        self.full=Label(self.root,font=("goudy old
style",15,'bold'),bg="white",bd=2,relief=GROOVE)

        self.full.place(x=750,y=280,width=150,height=50)

        self.per=Label(self.root,font=("goudy old
style",15,'bold'),bg="white",bd=2,relief=GROOVE)

        self.per.place(x=900,y=280,width=150,height=50)


        #======button delete=============

        btn_delete=Button(self.root,text='Delete',font=("goudy old
style",15,"bold"),bg="red",fg="white",cursor="hand2",command=self.delete).place(x=50
0,y=350,width=150,height=35)


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

    def search(self):

        con=sqlite3.connect(database="rms.db")

        cur=con.cursor()

        try:

            if self.var_search.get()=="":

                messagebox.showerror("Error","Roll no should be
required",parent=self.root)
```

```
            else:
                cur.execute(f"select * from result where roll=?",(self.var_search.get(),))
                row=cur.fetchone()
                if row!=None:
                    self.var_id=row[0]
                    self.roll.config(text=row[1])
                    self.name.config(text=row[2])
                    self.course.config(text=row[3])
                    self.marks.config(text=row[4])
                    self.full.config(text=row[5])
                    self.per.config(text=row[6])
                else:
                    messagebox.showerror("Error","No record found",parent=self.root)



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

    def clear(self):
        self.var_id=""
        self.roll.config(text="")
        self.name.config(text="")
        self.course.config(text="")
        self.marks.config(text="")
        self.full.config(text="")
        self.per.config(text="")
        self.var_search.set("")
```

```
    def delete(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            if self.var_id=="":
                messagebox.showerror("Error","Search Student result
first",parent=self.root)
            else:
                cur.execute("select * from result where rid=?",(self.var_id,))
                row=cur.fetchone()
                print(row)
                if row==None:
                    messagebox.showerror("Error","Invaild Student result
",parent=self.root)
                else:
                    op=messagebox.askyesno("Confirm","Do you really want to
delete?",parent=self.root)
                    if op==True:
                        cur.execute("delete from result where rid=?",(self.var_id,))
                        con.commit()
                        messagebox.showinfo("Delete","Result deleted
Successfully",parent=self.root)
                        self.clear()

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


class studentClass:
    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 Student Details",font=("goudy old
style",20,"bold"),bg="#033054",fg="white").place(x=10,y=15,width=1180,height=35)
        #=======Variables======
        self.var_roll=StringVar()
        self.var_name=StringVar()
        self.var_email=StringVar()
        self.var_gender=StringVar()
        self.var_dob=StringVar()
        self.var_contact=StringVar()
        self.var_course=StringVar()
        self.var_a_date=StringVar()
        self.var_state=StringVar()
        self.var_city=StringVar()
        self.var_pin=StringVar()



        #========Widgets ======================
        #=========column 1 ================
        lbl_roll=Label(self.root,text="Roll No.", font=("goudy old
style",15,'bold'),bg='white').place(x=10,y=60)
        lbl_Name=Label(self.root,text="Name", font=("goudy old
style",15,'bold'),bg='white').place(x=10,y=100)
        lbl_Email=Label(self.root,text="Email", font=("goudy old
style",15,'bold'),bg='white').place(x=10,y=140)
        lbl_gender=Label(self.root,text="Gender", font=("goudy old
style",15,'bold'),bg='white').place(x=10,y=180)
```

```
        lbl_state=Label(self.root,text="State", font=("goudy old
style",15,'bold'),bg='white').place(x=10,y=220)

        txt_state=Entry(self.root,textvariable=self.var_state, font=("goudy old
style",15,'bold'),bg='lightyellow').place(x=150,y=220,width=150)

        lbl_city=Label(self.root,text="City", font=("goudy old
style",15,'bold'),bg='white').place(x=310,y=220)

        txt_city=Entry(self.root,textvariable=self.var_city, font=("goudy old
style",15,'bold'),bg='lightyellow').place(x=380,y=220,width=100)

        lbl_pin=Label(self.root,text="Pin", font=("goudy old
style",15,'bold'),bg='white').place(x=500,y=220)

        txt_pin=Entry(self.root,textvariable=self.var_pin, font=("goudy old
style",15,'bold'),bg='lightyellow').place(x=560,y=220,width=120)

        lbl_address=Label(self.root,text="Address", font=("goudy old
style",15,'bold'),bg='white').place(x=10,y=260)




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


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

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

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

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

        self.txt_gender=ttk.Combobox(self.root,textvariable=self.var_gender,values=("Se
lect","Male","Female","Other"), font=("goudy old
style",15,'bold'),state='readonly',justify=CENTER)

        self.txt_gender.place(x=150,y=180,width=200)

        self.txt_gender.current(0)
```

```
        #============column 2=============

        lbl_dob=Label(self.root,text="D.O.B", font=("goudy old
style",15,'bold'),bg='white').place(x=360,y=60)

        lbl_contact=Label(self.root,text="Contact", font=("goudy old
style",15,'bold'),bg='white').place(x=360,y=100)

        lbl_addmission=Label(self.root,text="Addmission", font=("goudy old
style",15,'bold'),bg='white').place(x=360,y=140)

        lbl_course=Label(self.root,text="Course", font=("goudy old
style",15,'bold'),bg='white').place(x=360,y=180)

        #========Entry Fields2=======

        self.course_list=[]

        #function_call to update to the list

        self.fetch_course()

        txt_dob=Entry(self.root,textvariable=self.var_dob, font=("goudy old
style",15,'bold'),bg='lightyellow').place(x=480,y=60,width=200)

        txt_contact=Entry(self.root,textvariable=self.var_contact,font=("goudy old
style",15,'bold'),bg='lightyellow').place(x=480,y=100,width=200)

        txt_addmission=Entry(self.root,textvariable=self.var_a_date, font=("goudy old
style",15,'bold'),bg='lightyellow').place(x=480,y=140,width=200)

        self.txt_course=ttk.Combobox(self.root,textvariable=self.var_course,values=self.
course_list, font=("goudy old style",15,'bold'),state='readonly',justify=CENTER)

        self.txt_course.place(x=480,y=180,width=200)

        self.txt_course.set("Select")


        #===========Text Address=============


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

        self.txt_address.place(x=150,y=260,width=540,height=100)


        #=====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_roll=Label(self.root,text="Roll No. ", font=("goudy old
style",15,'bold'),bg='white').place(x=720,y=60)

        txt_search_roll=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).plac
e(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=("roll","name","email","gen
der","dob","contact","admission","course","state","city","pin","address"),xscrollcomma
nd=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("roll",text="Roll No")
self.CourseTable.heading("name",text="Name")
self.CourseTable.heading("email",text="Email")
self.CourseTable.heading("gender",text="Gender")
self.CourseTable.heading("dob",text="D.O.B")
self.CourseTable.heading("contact",text=" Contact")
self.CourseTable.heading("admission",text="Admission")
self.CourseTable.heading("course",text="Course")
self.CourseTable.heading("state",text="State")
self.CourseTable.heading("city",text="City")
self.CourseTable.heading("pin",text="PIN")
self.CourseTable.heading("address",text="Address")
self.CourseTable["show"]='headings'
self.CourseTable.column("roll",width=100)
self.CourseTable.column("name",width=100)
self.CourseTable.column("email",width=100)
self.CourseTable.column("gender",width=100)
self.CourseTable.column("dob",width=150)
self.CourseTable.column("contact",width=100)
self.CourseTable.column("admission",width=100)
self.CourseTable.column("course",width=100)
self.CourseTable.column("state",width=100)
self.CourseTable.column("city",width=100)
self.CourseTable.column("pin",width=100)
self.CourseTable.column("address",width=100)
self.CourseTable.pack(fill=BOTH,expand=1)
```

```
        self.CourseTable.bind("<ButtonRelease-1>",self.get_data)
        self.show()


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

    def clear(self):
        self.show()
        self.var_roll.set(""),
        self.var_name.set(""),
        self.var_email.set(""),
        self.var_gender.set("Select"),
        self.var_dob.set(""),
        self.var_contact.set(""),
        self.var_a_date.set(""),
        self.var_course.set("Select"),
        self.var_state.set(""),
        self.var_city.set(""),
        self.var_pin.set(""),
        self.txt_address.delete("1.0",END)
        self.txt_roll.config(state=NORMAL)
        self.var_search.set("")



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

```
            if self.var_roll.get()=="":
                messagebox.showerror("Error","Roll No. should be
required",parent=self.root)
            else:
                cur.execute("select * from student where roll=?",(self.var_roll.get(),))
                row=cur.fetchone()
                print(row)
                if row==None:
                    messagebox.showerror("Error","please select student 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 student where roll=?",(self.var_roll.get(),))
                        con.commit()
                        messagebox.showinfo("Delete","Student 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_roll.config(state='readonly')

        r=self.CourseTable.focus()
        content=self.CourseTable.item(r)
        row=content["values"]
```

```
        self.var_roll.set(row[0]),
        self.var_name.set(row[1]),
        self.var_email.set(row[2]),
        self.var_gender.set(row[3]),
        self.var_dob.set(row[4]),
        self.var_contact.set(row[5]),
        self.var_a_date.set(row[6]),
        self.var_course.set(row[7]),
        self.var_state.set(row[8]),
        self.var_city.set(row[9]),
        self.var_pin.set(row[10]),
        self.txt_address.delete("1.0",END)
        self.txt_address.insert(END,row[11])



    def add(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            if self.var_roll.get()=="":
                messagebox.showerror("Error","Roll No should be
required",parent=self.root)
            else:
                cur.execute("select * from student where roll=?",(self.var_roll.get(),))
                row=cur.fetchone()
                print(row)
                if row!=None:
                    messagebox.showerror("Error","Roll No already
avilable",parent=self.root)
                else:
```

```
                cur.execute("insert into
student(roll,name,email,gender,dob,contact,admission,course,state,city,pin,address)
values(?,?,?,?,?,?,?,?,?,?,?,?)",(
                    self.var_roll.get(),
                    self.var_name.get(),
                    self.var_email.get(),
                    self.var_gender.get(),
                    self.var_dob.get(),
                    self.var_contact.get(),
                    self.var_a_date.get(),
                    self.var_course.get(),
                    self.var_state.get(),
                    self.var_city.get(),
                    self.var_pin.get(),
                    self.txt_address.get("1.0",END)
                ))
                con.commit()
                messagebox.showinfo("Success","Student 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_roll.get()=="":
                messagebox.showerror("Error","Roll No should be
required",parent=self.root)
            else:
```

```
            cur.execute("select * from student where roll=?",(self.var_roll.get(),))
            row=cur.fetchone()
            print(row)
            if row==None:
                messagebox.showerror("Error","Select student from
list",parent=self.root)
            else:
                cur.execute("update student set
name=?,email=?,gender=?,dob=?,contact=?,admission=?,course=?,state=?,city=?,pin=?,a
ddress=? where roll=?",(

                    self.var_name.get(),
                    self.var_email.get(),
                    self.var_gender.get(),
                    self.var_dob.get(),
                    self.var_contact.get(),
                    self.var_a_date.get(),
                    self.var_course.get(),
                    self.var_state.get(),
                    self.var_city.get(),
                    self.var_pin.get(),
                    self.txt_address.get("1.0",END),
                    self.var_roll.get()
                ))
                con.commit()
                messagebox.showinfo("Success","Student 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 student ")
        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 fetch_course(self):
    con=sqlite3.connect(database="rms.db")
    cur=con.cursor()
    try:
        cur.execute("select name from course")
        rows=cur.fetchall()
        if len(rows)>0:
            for row in rows:
                self.course_list.append(row[0])
    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 student where roll=?",(self.var_search.get(),))
            row=cur.fetchone()
            if row!=None:
                self.CourseTable.delete(*self.CourseTable.get_children())
                self.CourseTable.insert('',END,values=row)
            else:
                messagebox.showerror("Error","No record found",parent=self.root)


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

if __name__ == "__main__":
    main()
```