

SOURCES CODE

B.TECH 3rd YR.(6th SEM.) 1st PROJECT

```
from tkinter import*
from tkinter import messagebox,ttk
import mysql.connector
import time

class EmployeeSystem:
    def __init__(self,root):

        self.root=root
        root.title("EMPLOYEE PAYROLL MANAGEMENT SYSTEM | BY SRIHARI KAUSHIK
BTECH. (CSE) (3rd YR.) ")
        root.geometry('1350x700')
        root.config(bg="white")
        title=Label(self.root,text="EMPLOYEE PAYROLL MANAGEMENT
SYSTEM",font=('times new roman',30,'bold'),bg='#262626',fg='white')
        title.place(x=0,y=0,relwidth=1)
        btn_emp=Button(root,text="ALL EMPLOYEE'S
DETAILS",command=self.employee_frame,font=('times new roman',10),bg='slate
blue',fg='blue')
        btn_emp.place(x=1100,y=14)

        #=====Frame1=====
        #=====Variables=====
        self.var_e_id=StringVar()
        self.var_designation=StringVar()
        self.var_name=StringVar()
        self.var_age=StringVar()
        self.var_gender=StringVar()
        self.var_email=StringVar()
        self.var_hr_location=StringVar()
        self.var_dob=StringVar()
        self.var_doj=StringVar()
        self.var_proof_id=StringVar()
```

```

self.var_contact=StringVar()
self.var_status=StringVar()
self.var_experience=StringVar()
self.var_address=StringVar()

Frame1=Frame(self.root,bd=5,relief=RIDGE,bg='white')
Frame1.place(x=10,y=70,width=715,height=560)

title2=Label(Frame1,text="Employee Details",font=('times new
romans',20),bg='lightgray',fg='black',anchor="w",padx=10)
title2.place(x=0,y=0,relwidth=1)

#=====ROW1=====
lbl_code=Label(Frame1,text="Emp.Code:",font=('times new
roman',16),bg='white',fg='black')
lbl_code.place(x=10,y=65)

self.txt_code=Entry(Frame1,font=('times new
roman',13),textvariable=self.var_e_id,bg='lightyellow',fg='black')
self.txt_code.place(x=150,y=65,width=200)

btn_search=Button(Frame1,text="Search",command=self.search,font=('times new
roman',16),bg='lightgreen',fg='black')
btn_search.place(x=390,y=55)

#=====ROW2=====
lbl_designation=Label(Frame1,text="Designation:",font=('times new
roman',16),bg='white',fg='black')
lbl_designation.place(x=10,y=115)

txt_designation=Entry(Frame1,font=('times new
roman',13),textvariable=self.var_designation,bg='lightyellow',fg='black')
txt_designation.place(x=150,y=115,width=200)

lbl_DOB=Label(Frame1,text="D.O.B:",font=('times new
roman',16),bg='white',fg='black')
lbl_DOB.place(x=400,y=115)

```

```

txt_DOB=Entry(Frame1,font=('times new
roman',13),textvariable=self.var_dob,bg='lightyellow',fg='black')
txt_DOB.place(x=500,y=115)

#=====ROW3=====

lbl_name=Label(Frame1,text="Name:",font=('times new
roman',16),bg='white',fg='black')
lbl_name.place(x=10,y=165)

txt_name=Entry(Frame1,font=('times new
roman',13),textvariable=self.var_name,bg='lightyellow',fg='black')
txt_name.place(x=150,y=165,width=200)

lbl_DOJ=Label(Frame1,text="D.O.J:",font=('times new
roman',16),bg='white',fg='black')
lbl_DOJ.place(x=400,y=165)

txt_DOJ=Entry(Frame1,font=('times new
roman',13),textvariable=self.var_doj,bg='lightyellow',fg='black')
txt_DOJ.place(x=500,y=165)

#=====ROW4=====

lbl_age=Label(Frame1,text="Age:",font=('times new
roman',16),bg='white',fg='black')
lbl_age.place(x=10,y=215)

txt_age=Entry(Frame1,font=('times new
roman',13),textvariable=self.var_age,bg='lightyellow',fg='black')
txt_age.place(x=150,y=215,width=200)

lbl_experience=Label(Frame1,text="Experience:",font=('times new
roman',16),bg='white',fg='black')
lbl_experience.place(x=380,y=215)

txt_experience=Entry(Frame1,font=('times new
roman',13),textvariable=self.var_experience,bg='lightyellow',fg='black')
txt_experience.place(x=500,y=215)

```

```

#=====ROW5=====

lbl_gender=Label(Frame1,text="Gender:",font=('times new
roman',16),bg='white',fg='black')
lbl_gender.place(x=10,y=265)

txt_gender=Entry(Frame1,font=('times new
roman',13),textvariable=self.var_gender,bg='lightyellow',fg='black')
txt_gender.place(x=150,y=265,width=200)

lbl_id=Label(Frame1,text="Proof ID:",font=('times new
roman',16),bg='white',fg='black')
lbl_id.place(x=380,y=265)

txt_id=Entry(Frame1,font=('times new
roman',13),textvariable=self.var_proof_id,bg='lightyellow',fg='black')
txt_id.place(x=500,y=265)

#=====ROW6=====

lbl_Email=Label(Frame1,text="Email:",font=('times new
roman',16),bg='white',fg='black')
lbl_Email.place(x=10,y=315)

txt_Email=Entry(Frame1,font=('times new
roman',13),textvariable=self.var_email,bg='lightyellow',fg='black')
txt_Email.place(x=150,y=315,width=200)

lbl_number=Label(Frame1,text="Contact No.:",font=('times new
roman',16),bg='white',fg='black')
lbl_number.place(x=380,y=315)

txt_number=Entry(Frame1,font=('times new
roman',13),textvariable=self.var_contact,bg='lightyellow',fg='black')
txt_number.place(x=500,y=315)

#=====ROW7=====

```

```

        lbl_address=Label(Frame1,text="Address:",font=('times new
roman',16),bg='white',fg='black')
        lbl_address.place(x=10,y=415)

        txt_address=Entry(Frame1,font=('times new
roman',13),textvariable=self.var_address,bg='lightyellow',fg='black')
        txt_address.place(x=150,y=415,width=535,height=60)

        title0=Label(Frame1,text="BY --SRIHARI KAUSHIK [B.TECH 3rd
YR.]",font=('times new
roman',20),bg='LightGoldenrod2',fg='black',anchor="w",padx=10).place(x=0,y=
514,relwidth=1)

#=====ROW8=====

        lbl_location=Label(Frame1,text="HR_Location:",font=('times new
roman',16),bg='white',fg='black')
        lbl_location.place(x=10,y=365)

        txt_location=Entry(Frame1,font=('times new
roman',13),textvariable=self.var_hr_location,bg='lightyellow',fg='black')
        txt_location.place(x=150,y=365,width=200)

        lbl_status=Label(Frame1,text="Status:",font=('times new
roman',16),bg='white',fg='black')
        lbl_status.place(x=400,y=365)

        txt_status=Entry(Frame1,font=('times new
roman',13),textvariable=self.var_status,bg='lightyellow',fg='black')
        txt_status.place(x=500,y=365)

#=====Frame2=====

        self.var_month=StringVar()
        self.var_year=StringVar()
        self.var_salary=StringVar()
        self.var_t_days=StringVar()
        self.var_absent=StringVar()
        self.var_medical=StringVar()

```

```

self.var_pf=StringVar()
self.var_convince=StringVar()
self.var_net_salary=StringVar()


Frame2=Frame(root,bd=3,relief=RIDGE,bg='white')
Frame2.place(x=730,y=70,width=540,height=300)


tile3=Label(Frame2,text="Employee Salary Details:",font=('times new
roman',20),bg='lightgray',fg='black',anchor="w",padx=10)
tile3.place(x=0,y=0,relwidth=1)


#=====2ROW1=====


lbl_Month=Label(Frame2,text="Month:",font=('times new
roman',13),bg='white',fg='black')
lbl_Month.place(x=10,y=60)


txt_Month=Entry(Frame2,font=('times new
roman',12),textvariable=self.var_month,bg='lightyellow',fg='black')
txt_Month.place(x=70,y=60,width=90)


lbl_year=Label(Frame2,text="Year:",font=('times new
roman',13),bg='white',fg='black')
lbl_year.place(x=200,y=60)


txt_year=Entry(Frame2,font=('times new
roman',12),textvariable=self.var_year,bg='lightyellow',fg='black')
txt_year.place(x=250,y=60,width=90)


lbl_salary=Label(Frame2,text="Basic:",font=('times new
roman',13),bg='white',fg='black')
lbl_salary.place(x=380,y=60)

```

```

txt_salary=Entry(Frame2,font=('times new
roman',12),textvariable=self.var_salary,bg='lightyellow',fg='black')
txt_salary.place(x=430,y=60,width=90)

#=====2ROW2=====

lbl_Days=Label(Frame2,text="Total Days:",font=('times new
roman',13),bg='white',fg='black')
lbl_Days.place(x=10,y=100)

txt_Days=Entry(Frame2,font=('times new
roman',13),textvariable=self.var_t_days,bg='lightyellow',fg='black')
txt_Days.place(x=100,y=100,width=150)

lbl_Absent=Label(Frame2,text="Days Absent:",font=('times new
roman',13),bg='white',fg='black')
lbl_Absent.place(x=265,y=100)

txt_Absent=Entry(Frame2,font=('times new
roman',13),textvariable=self.var_absent,bg='lightyellow',fg='black')
txt_Absent.place(x=370,y=100,width=150)

#=====2ROW3=====

lbl_medical=Label(Frame2,text="Medical:",font=('times new
roman',13),bg='white',fg='black')
lbl_medical.place(x=10,y=140)

txt_medical=Entry(Frame2,font=('times new
roman',13),textvariable=self.var_medical,bg='lightyellow',fg='black')
txt_medical.place(x=100,y=140,width=150)

lbl_pf=Label(Frame2,text="P.F.:",font=('times new
roman',13),bg='white',fg='black')
lbl_pf.place(x=320,y=140)

txt_pf=Entry(Frame2,font=('times new
roman',13),textvariable=self.var_pf,bg='lightyellow',fg='black')

```

```

txt_pf.place(x=370,y=140,width=150)

#=====2ROW4=====

lbl_convence=Label(Frame2,text="Convence:",font=('times new
roman',13),bg='white',fg='black')
lbl_convence.place(x=10,y=180)

txt_convence=Entry(Frame2,font=('times new
roman',13),textvariable=self.var_convence,bg='lightyellow',fg='black')
txt_convence.place(x=100,y=180,width=150)

lbl_net=Label(Frame2,text="Net Salary:",font=('times new
roman',13),bg='white',fg='black')
lbl_net.place(x=275,y=180)

txt_net=Entry(Frame2,font=('times new
roman',13),textvariable=self.var_net_salary,bg='lightyellow',fg='black')
txt_net.place(x=370,y=180,width=150)

btn_calculate=Button(Frame2,text="Calculate",command=self.calculate,font=('
times new roman',13),bg='lightblue',fg='black')
btn_calculate.place(x=138,y=240)

self.btn_save=Button(Frame2,text="Save",command=self.save,font=('times new
roman',13),bg='light green',fg='black')
self.btn_save.place(x=239,y=240)

self.btn_clear=Button(Frame2,text="Clear",command=self.clear,font=('times
new roman',13),bg='light blue',fg='black')
self.btn_clear.place(x=465,y=240)

```



```

self.btn_update=Button(Frame2,text="Update",state=NORMAL,command=self.update,font=('times new roman',13),bg='light green',fg='black')
    self.btn_update.place(x=305,y=240)


self.btn_delete=Button(Frame2,text="Delete",state=NORMAL,command=self.delete,font=('times new roman',13),bg='orange',fg='black')
    self.btn_delete.place(x=385,y=240)


#=====Frame3=====

Frame3=Frame(self.root,bd=5,relief=RIDGE,bg='white')
Frame3.place(x=730,y=380,width=540,height=250)


#=====Calculator Frame=====

self.var_txt=StringVar()

self.var_operator=''

def btn_click(num):
    self.var_operator=self.var_operator+str(num)
    self.var_txt.set(self.var_operator)

def result():
    res=str(eval(self.var_operator))
    self.var_txt.set(res)
    self.var_operator=''

def clear_cal():
    self.var_txt.set('')
    self.var_operator=''

cal_frame=Frame(Frame3,bg="antique white",bd=2,relief=RIDGE)
cal_frame.place(x=5,y=5,width=262,height=230)

```

```

text_result=Entry(cal_frame,bg='snow',textvariable=self.var_txt,font=("times
new roman",20,"bold"))
    text_result.place(x=5,y=5,width=250,height=40)

#=====Cal1=====

btn_7=Button(cal_frame,text="7",command=lambda:btn_click(7),font=("times
new roman",12,"bold"))
    btn_7.place(x=8,y=50,width='60',height='40')

btn_8=Button(cal_frame,text="8",command=lambda:btn_click(8),font=("times
new roman",12,"bold"))
    btn_8.place(x=68,y=50,width="60",height="40")

btn_9=Button(cal_frame,text="9",command=lambda:btn_click(9),font=("times
new roman",12,"bold"))
    btn_9.place(x=128,y=50,width="60",height="40")

btn_div=Button(cal_frame,text="/",command=lambda:btn_click("/"),font=("times
new roman",20,"bold"))
    btn_div.place(x=188,y=50,width="60",height="40")

#=====Cal2=====

btn_4=Button(cal_frame,text="4",command=lambda:btn_click(4),font=("times
new roman",12,"bold"))
    btn_4.place(x=8,y=92,width="60",height="40")

```

```

btn_5=Button(cal_frame,text="5",command=lambda:btn_click(5),font=("times
new roman",12,"bold"))
    btn_5.place(x=68,y=92,width="60",height="40")

btn_6=Button(cal_frame,text="6",command=lambda:btn_click(6),font=("times
new roman",12,"bold"))
    btn_6.place(x=128,y=92,width="60",height="40")

btn_mul=Button(cal_frame,text="*",command=lambda:btn_click("*"),font=("time
s new roman",20,"bold"))
    btn_mul.place(x=188,y=92,width="60",height="40")

#=====Cal3=====

btn_1=Button(cal_frame,text="1",command=lambda:btn_click(1),font=("times
new roman",12,"bold"))
    btn_1.place(x=8,y=135,width="60",height="40")

btn_2=Button(cal_frame,text="2",command=lambda:btn_click(2),font=("times
new roman",12,"bold"))
    btn_2.place(x=68,y=135,width="60",height="40")

btn_3=Button(cal_frame,text="3",command=lambda:btn_click(3),font=("times
new roman",12,"bold"))
    btn_3.place(x=128,y=135,width="60",height="40")

    btn_min=Button(cal_frame,text="-",command=lambda:btn_click("-
"),font=("times new roman",20,"bold"))
    btn_min.place(x=188,y=135,width="60",height="40")

#=====Cal4=====

```

```

btn_0=Button(cal_frame,text="0",command=lambda:btn_click(0),font=("times
new roman",12,"bold"))
    btn_0.place(x=8,y=177,width="60",height="40")

    btn_dot=Button(cal_frame,text="C",command=clear_cal,font=("times
new roman",12,"bold"))
    btn_dot.place(x=68,y=177,width="60",height="40")

btn_sum=Button(cal_frame,text="+",command=lambda:btn_click("+"),font=("time
s new roman",12,"bold"))
    btn_sum.place(x=128,y=177,width="60",height="40")

    btn_equal=Button(cal_frame,text="=",command=result,font=("times new
roman",20,"bold"))
    btn_equal.place(x=188,y=177,width="60",height="40")

#=====SALARY FRAME=====

sal_frame=Frame(Frame3,bg="white",bd=2,relief=RIDGE)
sal_frame.place(x=265,y=5,width=262,height=230)

title4=Label(sal_frame,text="Salary Receipt",font=('times new
roman',20),bg='lightgray',fg='black',anchor="w",padx=10)
title4.place(x=0,y=0,relwidth=1)

sal_frame2=Frame(sal_frame,bg="white",bd=2,relief=RIDGE)
sal_frame2.place(x=5,y=40,width=250,height=180)

self.sample=f'''\tINTERNATION SCHOOL\n Address:B-12
Kamlanagar,Floor-4

```

```

Employee ID\t\t: ID
Employee Name\t\t: -----

```

Emp. Designation\t\t: -----
Salary Of \t\t: MON/YYYY
Generated On\t\t: DD-MM-YYYY

Total Days\t\t: DD
Total Present\t\t: DD
Total Absent\t\t: DD
Convince\t\t: Rs.-----
Medical\t\t: Rs.-----
PF \t\t: Rs.-----
Gross Payment\t\t: Rs.-----
Net Salary\t\t: Rs.-----

This is computer generated slip,
not required any signature''

```
scroll_y=Scrollbar(sal_frame2,orient=VERTICAL)
scroll_y.pack(side=RIGHT,fill=Y)

self.txt_salary_receipt=Text(sal_frame2,font=("times new
roman",11),bg="lightyellow",yscrollcommand=scroll_y.set)
self.txt_salary_receipt.pack(side=LEFT,fill="both",expand=1)

scroll_y.config(command=self.txt_salary_receipt.yview)
self.txt_salary_receipt.insert(END,self.sample)

#=====ALL FUNCTIONS STARTS=====

def search(self):
    try:
        con=mysql.connector.connect(host='localhost',user='root',
                                    password='kaushik@',db='project')
```

```

c=con.cursor()

c.execute('SELECT * FROM employee_data WHERE
e_id=%s',(str(self.var_e_id.get()),))

row=c.fetchone()

if row is not None:
    self.var_e_id.set(row[0])
    self.var_designation.set(row[1])
    self.var_name.set(row[2])
    self.var_age.set(row[3])
    self.var_gender.set(row[4])
    self.var_email.set(row[5])
    self.var_hr_location.set(row[6])
    self.var_doj.set(row[7])
    self.var_dob.set(row[8])
    self.var_experience.set(row[9])
    self.var_proof_id.set(row[10])
    self.var_contact.set(row[11])
    self.var_status.set(row[12])
    #self.var_address.delete('1.0',END)
    #self.var_address.insert(END,row[13])

    self.var_month.set(row[14])
    self.var_year.set(row[15])
    self.var_salary.set(row[16])
    self.var_t_days.set(row[17])
    self.var_absent.set(row[18])
    self.var_medical.set(row[19])
    self.var_pf.set(row[20])
    self.var_convince.set(row[21])
    self.var_net_salary.set(row[22])

    file=open('Salary_receipt/'+str(row[23]),'r')

    self.txt_salary_receipt.delete('1.0',END)

```

```

        for i in file:
            self.txt_salary_receipt.read(END,i)
            #self.txt_salary_receipt.insert(END,i)
        file.close()

        self.btn_save.config(state=NORMAL)
        self.btn_update.config(state=NORMAL)
        self.btn_delete.config(state=NORMAL)
        self.txt_code.config(state='readonly')

    else:
        messagebox.showerror("Error",'Invalid Employee ID,TRY AGAIN
with another ID',parent=self.root)

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

def clear(self):
    self.btn_save.config(state=NORMAL)
    self.btn_update.config(state=NORMAL)
    self.btn_delete.config(state=NORMAL)
    self.txt_code.config(state=NORMAL)

    entry_widgets=(self.var_e_id, self.var_designation, self.var_name,
self.var_age, self.var_gender,
                    self.var_email, self.var_hr_location, self.var_dob,
self.var_doj, self.var_proof_id,
                    self.var_contact, self.var_status,
self.var_experience, self.var_address, self.var_month,
                    self.var_year, self.var_salary, self.var_t_days,
self.var_absent, self.var_medical,
                    self.var_pf, self.var_convence, self.var_net_salary)

    for entry in entry_widgets:
        entry.set("")

```

```

        self.txt_salary_receipt.delete("1.0", "end")
        self.txt_salary_receipt.insert(END, self.sample)

    def delete(self):
        if self.var_e_id.get()=='':
            messagebox.showerror("Error", 'EMPLOYEE ID IS REQUIRED')
        else:
            try:

con=mysql.connector.connect(host='localhost',user='root',password='kaushik@
',db='project')

                c=con.cursor()
                c.execute('SELECT * FROM employee_data WHERE
e_id=%s',(str(self.var_e_id.get()),))
                row=c.fetchone()
                if row==None:
                    messagebox.showerror("Error", 'Invalid Employee ID, try
again with another ID', parent=self.root)
                else:
                    op=messagebox.askyesno("CONFIRM", 'DO YOU REALLY WANT TO
DELETE?')

                    if op==True:
                        c.execute("DELETE FROM employee_data WHERE
e_id=%s",(str(self.var_e_id.get()),))
                        con.commit()
                        messagebox.showinfo('SUCCESS', 'DATA DELETED
SUCCESSFULLY')

                        self.clear()

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

    def calculate(self):
        if (self.var_month.get()==' ' or self.var_year.get()==' ' or
            self.var_salary.get()==' ' or self.var_t_days.get()==' ' or
            self.var_absent.get()==' ' or self.var_e_id.get()==' ' or

```



```

        self.var_name.get()==' ' or self.var_status.get()==' '):
            messagebox.showerror('Error','All fields are required')

    else:
        per_day=int(self.var_salary.get())/int(self.var_t_days.get())
        work_days=int(self.var_t_days.get()) -
int(self.var_absent.get())
        sal_=(per_day) * (work_days)
        deduct=int(self.var_medical.get())+int(self.var_pf.get())
        addition=int(self.var_convence.get())
        net_sal=(sal_-deduct)+(addition)
        self.var_net_salary.set(str(round(net_sal,2)))

```

#=====UPDATE THE RECEIPT=====

```

        new_sample=f'''\tINTERNATIONAL SCHOOL\n Address:B-12
Kamlanagar,Floor-4

```

```

-----

Employee ID\t\t: {self.var_e_id.get()}
Employee Name\t\t: {self.var_name.get()}
Emp. Designation\t\t: {self.var_designation.get()}
Salary Of \t\t: {self.var_month.get()}-{self.var_year.get()}
Generated On\t\t: {str(time.strftime("%d-%m-%y"))}

-----

```

```

Total Days\t\t: {self.var_t_days.get()}
Total Present\t\t: {str(int(self.var_t_days.get())-
int(self.var_absent.get()))}
Total Absent\t\t: {self.var_absent.get()}
Convence\t\t: Rs. {self.var_convence.get()}
Medical\t\t : Rs. {self.var_medical.get()}
PF \t\t      : Rs. {self.var_pf.get()}
Gross Payment\t\t: Rs. {self.var_salary.get()}
Net Salary\t\t: Rs. {self.var_net_salary.get()}

```

This is computer generated slip,
not required any signature
, , ,

```
self.txt_salary_receipt.delete('1.0',END)
self.txt_salary_receipt.insert(END,new_sample)
```

```
def save(self):
    if (self.var_e_id.get()==' ' or self.var_name.get()==' ' or
self.var_designation.get()==''):
        messagebox.showerror("Error","ALL EMPLOYEE DETAILS ARE
REQUIRED")

    else:
        try:
            #connection=mysql.connector.connect(host='localhost',
user='root',#password='kaushik@')

            #c=connection.cursor()

            #query_1="CREATE DATABASE project"
            #c.execute(query_1)
            #con.commit()
            #print("DATABASE CREATED")

            #connection=mysql.connector.connect(host='localhost',
user='root', #password='kaushik@',db='project')

            #c=connection.cursor()

            #query_2=("CREATE TABLE employee_data (E_ID INT(10) NOT
NULL PRIMARY KEY,DESIGNATION TINYTEXT,NAME TINYTEXT,AGE TINYTEXT,GENDER
TINYTEXT,EMAIL TINYTEXT,HR_LOCATION TINYTEXT,DOJ TINYTEXT,DOB
```

```
TINYTEXT,EXPERIENCE TINYTEXT,PROOF_ID TINYTEXT,CONTACT TINYTEXT,STATUS
TINYTEXT,ADDRESS TINYTEXT,MONTH TINYTEXT,YEAR TINYTEXT,BASIC_SALARY
TINYTEXT,T_DAYS TINYTEXT,ABSENT_DAYS TINYTEXT,MEDICAL TINYTEXT,PF
TINYTEXT,CONVENCE TINYTEXT,NET_SALARY TINYTEXT,SALARY_RECEIPT TEXT)")
```

```
#c.execute(query_2)
#connection.commit
#print("Table Created")
```

```
connection=mysql.connector.connect(host='localhost',
user='root',
```

```
password='kaushik@',db='project')
```

```
c=connection.cursor()
c.execute("SELECT * FROM employee_data WHERE
e_id=%s",(str(self.var_e_id.get()),))
row=c.fetchone()
```

```
if row!=None:
    messagebox.showerror("ERROR","THIS EMPLOYEE ID IS
ALREADY AVAILABLE IN OUR RECORDS")
```

```
else:
    e_id=self.var_e_id.get()
    designation=self.var_designation.get()
    name=self.var_name.get()
    age=self.var_age.get()
    gender=self.var_gender.get()
    email=self.var_email.get()
    location=self.var_hr_location.get()
    doj=self.var_doj.get()
    dob=self.var_dob.get()
    experience=self.var_experience.get()
    proof=self.var_proof_id.get()
    contact=self.var_contact.get()
    status=self.var_status.get()
    address=self.var_address.get()
```

```

        month=self.var_month.get()
        year=self.var_year.get()
        basic=self.var_salary.get()
        total=self.var_t_days.get()
        absent=self.var_absent.get()
        medical=self.var_medical.get()
        pf=self.var_pf.get()
        convence=self.var_convence.get()
        net=self.var_net_salary.get()
        #receipt=self.var_e_id.get().txt

receipt="Salary_receipt/"+str(self.var_e_id.get()+'.txt')

        insert_query= "INSERT INTO employee_data
(e_id,designation,name,age,gender,email,hr_location,doj,dob,experience,proof_id,contact,status,address,month,year,basic_salary,t_days,absent_days,medical,pf,convence,net_salary,salary_receipt) VALUES
(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"

vals=(e_id,designation,name,age,gender,email,location,doj,dob,experience,proof,contact,status,address,month,year,basic,total,absent,medical,pf,convence,net,receipt)

        c.execute(insert_query,vals)

        connection.commit()
        connection.close()

file=open("Salary_receipt/"+str(self.var_e_id.get()+'.txt'),'w')

        file.write(self.txt_salary_receipt.get('1.0',END))

        file.close()

```

```

        messagebox.showinfo("Success", "RECORD ADDED
SUCCESSFULLY")

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

def update(self):
    if self.var_e_id.get()==" " and self.var_name.get()==" ":
        messagebox.showerror("ERROR", "EMPLOYEE ID AND NAME ARE
REQUIRED")

    else:
        try:
            connection=mysql.connector.connect(host='localhost',
user='root',

password='kaushik@',db='project')
            c=connection.cursor()

            #c.execute("update employee_data
set=", (str(self.var_e_id.get()),))

            c.execute("SELECT * FROM employee_data WHERE
e_id=%s", (str(self.var_e_id.get()),))

            row=c.fetchone()

            if row==None:
                messagebox.showerror("ERROR", "THIS EMPLOYEE ID IS
INVALID, TRY AGAIN WITH ANOTHER ID")

            else:
                e_id=self.var_e_id.get()
                designation=self.var_designation.get()
                name=self.var_name.get()

```

```

age=self.var_age.get()
gender=self.var_gender.get()
email=self.var_email.get()
location=self.var_hr_location.get()
doj=self.var_doj.get()
dob=self.var_dob.get()
experience=self.var_experience.get()
proof=self.var_proof_id.get()
contact=self.var_contact.get()
status=self.var_status.get()
address=self.var_address.get()

month=self.var_month.get()
year=self.var_year.get()
basic=self.var_salary.get()
total=self.var_t_days.get()
absent=self.var_absent.get()
medical=self.var_medical.get()
pf=self.var_pf.get()
convence=self.var_convence.get()
net=self.var_net_salary.get()
receipt=str(self.var_e_id.get()) + ".txt"

update_query=("UPDATE employee_data SET
designation=%s,name=%s,age=%s,gender=%s,email=%s,hr_loaction=%s,doj=%s,dob=
%s,experience=%s,proof_id=%s,contact=%s,status=%s,address=%s,month=%s,year=
%s,basic_salary=%s,t_days=%s,absent_days=%s,medical=%s,pf=%s,convence=%s,ne
t_salary=%s,salary_receipt=%s WHERE e_id=%s")

vals=(designation,name,age,gender,email,location,doj,dob,experience,proof,c
ontact,status,address,month,year,basic,total,absent,medical,pf,convence,net
,e_id)

c.execute(update_query,vals)

connection.commit()

```

```

        #connect.close()

        file=open('Salary_receipt/'+receipt,'w')

        file.write(self.txt_salary_receipt.get('1.0',END))

        file.close()

        messagebox.showinfo("Success","RECORD UPDATED
SUCCESSFULLY")

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

def show(self):
    try:
        connection=mysql.connector.connect(host='localhost',
user='root',

password='kaushik@',db='project')
        c=connection.cursor()

        c.execute("SELECT * FROM employee_data")

        row=c.fetchall()

        self.employee_tree.delete(*self.employee_tree.get_children())

        for row in row:
            self.employee_tree.insert('',END,values=row)

    except Exception as ex:
        messagebox.showerror("ERROR",f'ERROR DUE TO:{str(ex)}')

```

```

def employee_frame(self):
    self.root2=Toplevel(self.root)
    self.root2.title("EMPLOYEE PAYROLL MANAGEMENT SYSTEM | BY SRIHARI
KAUSHIK BTECH. (CSE) (3rd YR.)")
    self.root2.geometry("900x500+200+60")
    self.root2.config(bg="white")

    title=Label(self.root2,text="Employee Payroll Management
System",font=('times new roman',30,'bold'),bg='lightblue',fg='black')
    title.pack(side=TOP,fill=X)

    self.root2.focus_force()

    scrolly=Scrollbar(self.root2,orient=VERTICAL)
    scrolly.pack(side=RIGHT,fill=Y)

    scrollx=Scrollbar(self.root2,orient=HORIZONTAL)
    scrollx.pack(side=BOTTOM,fill=X)


self.employee_tree=ttk.Treeview(self.root2,column=('e_id','designation','na
me','age','gender','email','hr_location','doj','dob','experience',

'proof_id','contact','status','address','month','year','basic_salary','t_da
ys','absent_days','medical',

'pf','convenience','net_salary','salary_receipt'),
                                yscrollcommand=scrolly.set,
                                xscrollcommand=scrollx.set)

self.employee_tree.heading('e_id',text='EMPLOYEE ID')
self.employee_tree.heading('designation',text='DESIGNATION')
self.employee_tree.heading('name',text='NAME')

```



```

self.employee_tree.heading('age',text='AGE')
self.employee_tree.heading('gender',text='GENDER')
self.employee_tree.heading('email',text='EMAIL')
self.employee_tree.heading('hr_location',text='HR_LOCATION')
self.employee_tree.heading('doj',text='DOJ')
self.employee_tree.heading('dob',text='DOB')
self.employee_tree.heading('experience',text='EXPERIENCE')
self.employee_tree.heading('proof_id',text='PROOF_ID')
self.employee_tree.heading('contact',text='CONTACT')
self.employee_tree.heading('status',text='STATUS')
self.employee_tree.heading('address',text='ADDRESS')
self.employee_tree.heading('month',text='MONTH')
self.employee_tree.heading('year',text='YEAR')
self.employee_tree.heading('basic_salary',text='BASIC SALARY')
self.employee_tree.heading('t_days',text='TOTAL DAYS')
self.employee_tree.heading('absent_days',text='ABSENT DAYS')
self.employee_tree.heading('medical',text='MEDICAL')
self.employee_tree.heading('pf',text='PF')
self.employee_tree.heading('convenience',text='CONVENIENCE')
self.employee_tree.heading('net_salary',text='NET SALARY')
self.employee_tree.heading('salary_receipt',text='SALARY RECEIPT')
self.employee_tree ['show']='headings'

```

```

self.employee_tree.column('e_id',width=100)
self.employee_tree.column('designation',width=100)
self.employee_tree.column('name',width=200)
self.employee_tree.column('age',width=100)
self.employee_tree.column('gender',width=100)
self.employee_tree.column('email',width=150)
self.employee_tree.column('hr_location',width=100)
self.employee_tree.column('doj',width=100)
self.employee_tree.column('dob',width=100)
self.employee_tree.column('experience',width=100)
self.employee_tree.column('proof_id',width=100)

```

```

self.employee_tree.column('contact',width=100)
self.employee_tree.column('status',width=100)
self.employee_tree.column('address',width=300)
self.employee_tree.column('month',width=100)
self.employee_tree.column('year',width=100)
self.employee_tree.column('basic_salary',width=100)
self.employee_tree.column('t_days',width=100)
self.employee_tree.column('absent_days',width=100)
self.employee_tree.column('medical',width=100)
self.employee_tree.column('pf',width=100)
self.employee_tree.column('convence',width=100)
self.employee_tree.column('net_salary',width=100)
self.employee_tree.column('salary_receipt',width=100)

scrollx.config(command=self.employee_tree.xview)
scrolly.config(command=self.employee_tree.yview)

self.employee_tree.pack(fill=BOTH,expand=1)
self.show()
self.root2.mainloop()

```

```

root=Tk()
obj= EmployeeSystem(root)
root.mainloop()

```

OUTPUTS

EMPLOYEE PAYROLL MANAGEMENT SYSTEM | BY SRIHARI KAUSHIK BTECH.(3rd YR.)

ALL EMPLOYEE'S DETAILS

Employee Details

Emp.Code:

Search

Designation:

D.O.B:

Name:

D.O.J:

Age:

Experience:

Gender:

Proof ID:

Email:

Contact No.:

HR_Location:

Status:

Address:

Employee Salary Details:

Month:

Year:

Basic:

Total Days:

Days Absent:

Medical:

P.F.:

Convenence:

Net Salary:

Calculate

Save

Update

Delete

Clear

789/

456*

123-

0C+ =

Salary Receipt

INTERNATION SCHOOL
Address:B-12 Kamlanagar,Floor-4

Employee ID : ID
Employee Name : ----
Emp. Designation : ----
Salary Of : MON/YYYY
Generated On : DD-MM-YYYY

Total Days : DD

BY --SRIHARI KAUSHIK [B.TECH 3rd YR.]

```
mysql> DESC employee_data;
```

Field	Type	Null	Key	Default	Extra
E_ID	int(10)	NO	PRI	NULL	
DESIGNATION	tinytext	YES		NULL	
NAME	tinytext	YES		NULL	
AGE	tinytext	YES		NULL	
GENDER	tinytext	YES		NULL	
EMAIL	tinytext	YES		NULL	
HR_LOCATION	tinytext	YES		NULL	
DOJ	tinytext	YES		NULL	
DOB	tinytext	YES		NULL	
EXPERIENCE	tinytext	YES		NULL	
PROOF_ID	tinytext	YES		NULL	
CONTACT	tinytext	YES		NULL	
STATUS	tinytext	YES		NULL	
ADDRESS	tinytext	YES		NULL	
MONTH	tinytext	YES		NULL	
YEAR	tinytext	YES		NULL	
BASIC_SALARY	tinytext	YES		NULL	
T_DAYS	tinytext	YES		NULL	
ABSENT_DAYS	tinytext	YES		NULL	
MEDICAL	tinytext	YES		NULL	
PF	tinytext	YES	NULL		
CONVENCE	tinytext	YES	NULL		
NET_SALARY	tinytext	YES	NULL		
SALARY_RECEIPT	text	YES	NULL		

BY --SRIHARI KAUSHIK [B.TECH 3rd YR.]

BY --SRIHARI KAUSHIK [B.TECH 3rd YR.]

BY --SRIHARI KAUSHIK [B.TECH 3rd YR.]

BY --SRIHARI KAUSHIK [B.TECH 3rd YR.]

INTERNATIONAL SCHOOL
Address:B-12 Kamlanagar

Employee ID : 117
Employee Name : KALPANA
Emp. Designation : COMPUTER PGT
Salary Of : NOV-2021
Generated On : 09-04-24

Total Days : 30
Total Present : 30
Total Absent : 0
Convence : Rs. 5000
Medical : Rs. 2400
PF : Rs. 4500
Gross Payment : Rs. 45000
Net Salary : Rs. 43100.0

This is computer generated slip,
not required any signature

INTERNATIONAL SCHOOL
Address:B-12 Kamlanagar

Employee ID : 118
Employee Name : MUKUL
Emp. Designation : ENGLISH PGT
Salary Of : JAN-2022
Generated On : 09-04-24

Total Days : 30
Total Present : 22
Total Absent : 8
Convence : Rs. 5000
Medical : Rs. 2400
PF : Rs. 4500
Gross Payment : Rs. 35000
Net Salary : Rs. 23766.67

This is computer generated slip,
not required any signature

Employee Details

Emp.Code:

Search

Designation:

D.O.B:

Name:

D.O.J:

Age:

Experience:

Gender:

Proof ID:

Email:

Contact No.:

HR_Location:

Status:

Address:

BY --SRIHARI KAUSHIK [B.TECH 3rd YR.]

Employee Salary Details:

Month:

Year:

Basic:

Total Days:

Days Absent:

Medical:

P.F.:

Convenence:

Net Salary:

Calculate

Save

Update

Delete

Clear

789/

456*

123-

0C+=

Salary Receipt

INTERNATION SCHOOL
Address:B-12 Kamlanagar,Floor-4

Employee ID : ID
Employee Name : ----
Emp. Designation : ----
Salary Of : MON/YYYY
Generated On : DD-MM-YYYY

Total Days : DD

789/

456*

123-

0C+=

4*9

789/

456*

123-

0C+=

36

789/

456*

123-

0C+=

BY --SRIHARI KAUSHIK [B.TECH 3rd YR.]

BY --SRIHARI KAUSHIK [B.TECH 3rd YR.]

BY --SRIHARI KAUSHIK [B.TECH 3rd YR.]

BY --SRIHARI KAUSHIK [B.TECH 3rd YR.]

●●●

EMPLOYEE PAYROLL MANAGEMENT SYSTEM | BY SRIHARI KAUSHIK BTECH.(CSE)(3rd YR.)

Employee Payroll Management System

EMPLOYEE ID	DESIGNATION	NAME	AGE	GENDER	EMAIL	HR_LOCATION	
102	Sales Associat	Parth Sharma	21	Male	ps23@gmail.com	Gurgaon	24/0
117	COMPUTER PG	KALPANA	33	FEMALE	kalpana16@gmail.com	DELHI	01/0
118	ENGLISH PGT	MUKUL	30	MALE	mukul11@gmail.com	DELHI	01/0

●●●

EMPLOYEE PAYROLL MANAGEMENT SYSTEM | BY SRIHARI KAUSHIK BTECH.(CSE)(3rd YR.)

Employee Payroll Management System

DOJ	DOB	EXPERIENCE	PROOF_ID	CONTACT	STATUS	ADDRESS
24/04/2020	10/06/2007	1yrs.	0002	8765432109	Permanent	Sec.-100, Noida-110056
01/04/2020	16/06/1990	5 YR.	000117	1234567890	PERMANENT	Sec.-42, Delhi-110024
01/04/2020	11/10/1995	2YR.	000118	0987654321	PROBATION	Sec.-56, Delhi-110065

●●●

EMPLOYEE PAYROLL MANAGEMENT SYSTEM | BY SRIHARI KAUSHIK BTECH.(CSE)(3rd YR.)

Employee Payroll Management System

MONTH	YEAR	BASIC SALARY	TOTAL DAYS	ABSENT DAYS	MEDICAL	PF	CONVENCE	NET SALARY
OCT	2023	46000	30	4	1500	3600	1600	36366.67
NOV	2021	45000	30	0	2400	4500	5000	43100.0
JAN	2022	35000	30	8	2400	4500	5000	23766.67

●●●

EMPLOYEE PAYROLL MANAGEMENT SYSTEM | BY SRIHARI KAUSHIK BTECH.(CSE)(3rd YR.)

Employee Payroll Management System

YEAR	BASIC SALARY	TOTAL DAYS	ABSENT DAYS	MEDICAL	PF	CONVENCE	NET SALARY	SALARY RECEIP
23	46000	30	4	1500	3600	1600	36366.67	102.txt
21	45000	30	0	2400	4500	5000	43100.0	Salary_receip
22	35000	30	8	2400	4500	5000	23766.67	Salary_receip

CONCLUTION

In conclusion, this employee payroll system offers an efficient solution for managing employee data and payroll processing. By leveraging Python and MySQL, it provides a user-friendly interface for storing employee information, calculating salaries, and generating pay slips. This system streamlines HR tasks, making it easier to maintain records, update information, and ensure accurate compensation for employees.

The code's modular design allows for easy customization and scalability, making it adaptable to various organizational needs. It showcases the power of open-source technologies in building robust business applications.

As startups and small organizations strive to enhance efficiency and reduce manual processes, this payroll system serves as a valuable tool. It not only simplifies HR operations but also minimizes errors, ensuring that employees are compensated accurately and on time.

In the ever-evolving world of business and technology, tools like this employee payroll system play a vital role in supporting modern HR practices and contributing to the success of organizations.

FUTURE SCOPE

1. Integration with External Systems: One potential future enhancement is to integrate the payroll system with other HR and finance software, such as attendance management, tax calculation, and accounting systems. This integration can streamline data flow, reducing manual data entry and potential errors.

2. Security Enhancements: Implement advanced security features to protect sensitive employee data. This may involve role-based access control and encryption to ensure data privacy and compliance with data protection regulations.

3. Mobile Application: Develop a mobile application to allow HR personnel and employees to access and update their information on the go. This can improve accessibility and convenience.

4. Cloud-Based Deployment: Offer a cloud-based version of the software, allowing organizations to access and manage payroll data from anywhere, while also facilitating automatic updates and backups.

5. Localization and Multilingual Support: Extend the system to support multiple languages and comply with local tax and labor laws in different regions, making it suitable for global organizations.

As businesses evolve and technology advances, this payroll system can stay relevant and valuable by incorporating these and some other future enhancements to meet the growing demands of modern HR and payroll management.

ADVANTAGES

1. **Efficient Employee Data Management:** The code allows for the centralized and organized storage of employee details, making it easy to update and retrieve information. This efficiency reduces the administrative burden on HR departments.
2. **Accurate Salary Calculation:** The code automates the process of salary calculation, reducing the chances of errors. This leads to precise salary disbursements and minimizes disputes.
3. **Streamlined Payroll Processing:** With this code, generating salary receipts becomes a straightforward task. This streamlining of payroll processing saves time and ensures that employees are paid on time.
4. **User-Friendly Interface:** The system offers a user-friendly interface that simplifies interactions for HR personnel and administrators. This means minimal training is required for users to operate the system effectively.
5. **Customization:** The code can be customized to fit the specific needs and requirements of an organization. This adaptability ensures it aligns with unique payroll processes.

BIBLIOGRAPHY

- . Computer Science with Preeti Arora
- . www.stackoverflow.com
- . www.geeksforfeeks.org
- . www.github.com
- . www.youtube.com
- . www.useblackbox.io