INDEX

INTRODUCTION

The project entitled "Employee Payroll Management System" is a project of MCA 3rd Semester students. It is a small software package which is helpful in the areas of Payment. This program package include all the activities in which salary is managed. The activities like save employee data, update and delete employee data, etc are included.

In terms of security this software is totally secured as it doesn't give access to unauthorized users as the username and password facility is provided. Hence, this software is helpful for the employee payroll management system teams who want to improve the efficiency of the management and want to be the competitive in this day to day changing throw world. This project is also a small step towards development of big professional software.

In the existing system, most of the records are maintained on paper, it becomes very inconvenient to modify the data. In the existing system, here is a possibility that the same data in different registers may have different values which means the entries of the same data do not match. This inconsistent state does not supply the concrete information which poses a problem in the case information related to particular search record.

Our project is very useful. User is no longer required to check his register in search of records, as now it can be searched over the software by choosing some options. The user need not to type in most of the information. User just required to enter the desired options. On the whole it liberates the user from keeping lengthy manual records.

TOOLS AND TECHNOLOGY

DATABASE:-

This module will keep the lists and stock details of all the furniture available ,all the details about the furniture and all the information about the user , their orders and their payments details.

What is Database?

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.

Other kinds of data stores can be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those types of systems.

So nowadays, we use relational database management system (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as foreign keys.

A Relational Database Management System (RDBMS) is a software that:

Enables you to implement a database with tables, columns and indexes. Guarantees the Referential Integrity between rows of various tables. Updates the indexes automatically. Interprets an SQL query and combines information from various tables.

MySQL Database:-

MySQL is a fast, easy-to-use RDBMS being used for many small and big business. MySQl is developed, marketed, and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons:

- MySQL is released under an open-source license. So you have nothing to pay to use it.
- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, Python,etc.
- MySQL works very quickly and works well even with large data sets.
- MySQL is very friendly to PHP, the most appreciated language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes(TB).
- MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

FEATURES OF MYSQL:-

- MySQL follows ANSI SQL 99, the standard SQL>
- Cross platform
- RDBMS features like stored procedures, triggers, cursor, views when can be updated.
- Full text indexing and searching.
- Partial Unicode support.

HARDWARE AND SOFTWARE

Hardware Requirement:-

- The hardware used for the development of the project is:

- Processor : intel(R) Core(TM) i3-6006U CPU @ 2.00GHz 2.00 GHz

- RAM : 4.00 GB (3.87 GB usable)
- Hard Disk : 1TB
- Monitor : LED
- Keyboard/Mouse : Optical

Software Requirement:-

- The software used for the development of the project is:

- Operating System : Windows 10
- Front_End : Python
- Back_End : MySQL
- IDE : Visual Studio Code
- Tools used : IDLE Shell 3.10.6

VISUAL STUDIO CODE

Visual Studio (VS) Code is an open-source code editor primarily used to correct and repair cloud and web applications coding errors. VS Code is developed by Microsoft and supports the macOS, Linux, and Windows operating systems. VS Code's tools can be used to enhance the functionality of any written code. Based on the Electron framework, VS Code employs the same editor component used in Azure DevOps. By incorporating multiple FTP extensions, Users can sync code between the server and the editor without having to download extra software.

HOW VISUAL STUDIO CODE WORKS

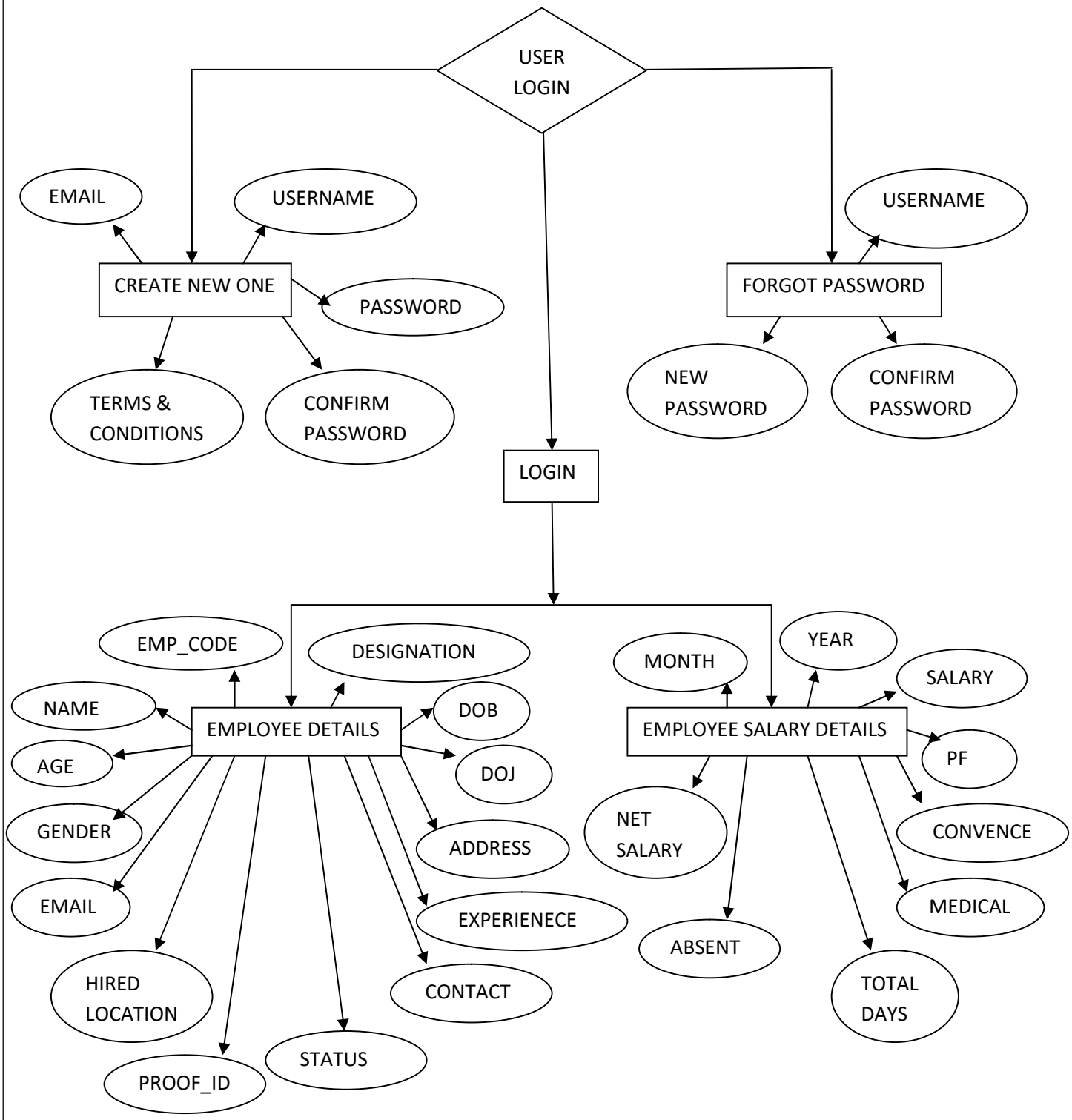
Once downloaded and installed, VS Code launches, identifies the programming language used, and what specific features to implement. The developer can then explore or search options to find and correct coding errors. Major editing is done via the extension option, while source control allows utilization of the Git repository. Users can customize and fine-tune their code using the VS Code's Interactive Editor Playground before deploying their project.

BENEFITS OF VISUAL STUDIO CODE

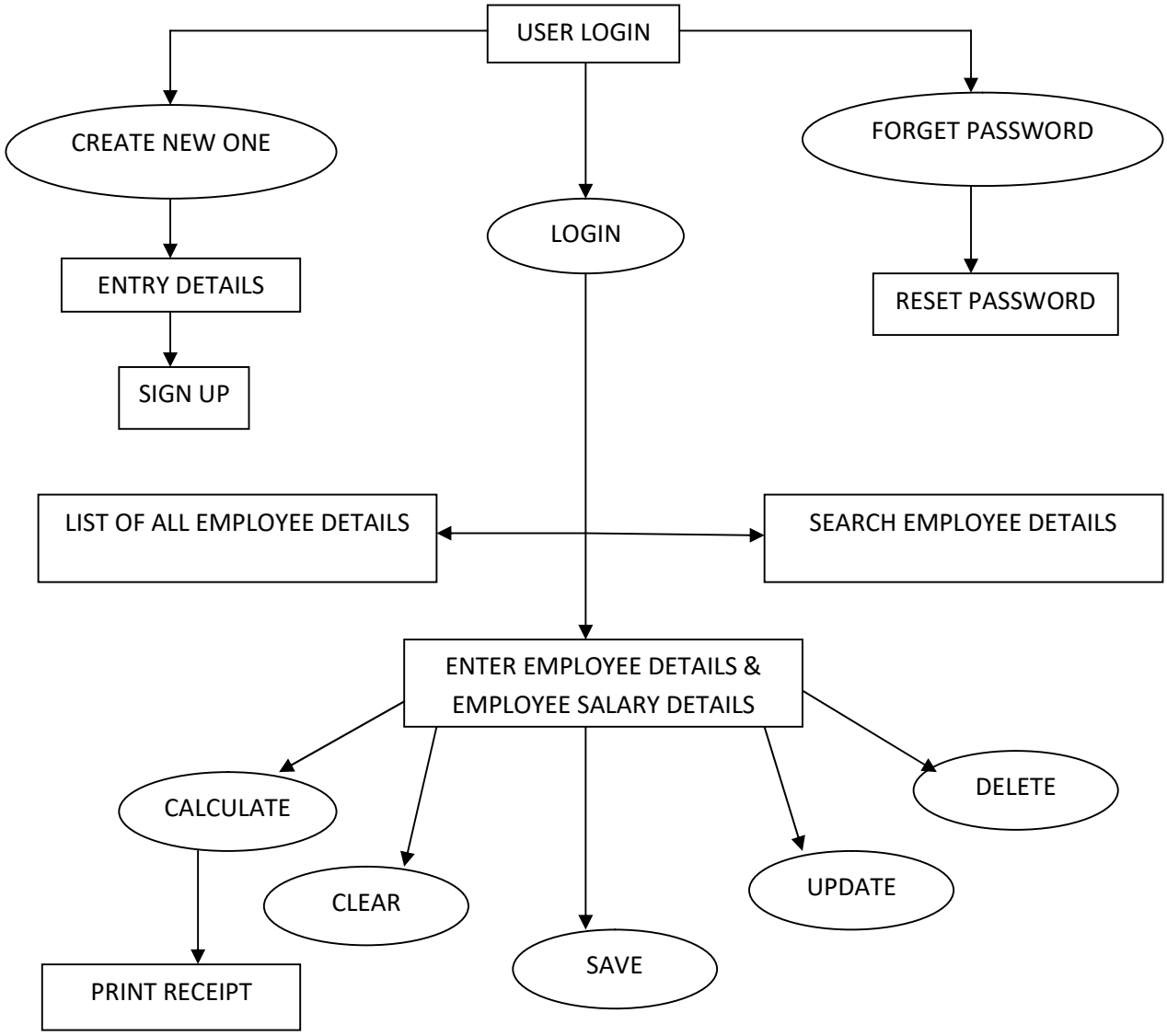
Programmers can swiftly navigate between multiple tools to make the necessary corrections to their code. Aside from its easy-to-use customization options, VS Code also offers keyboard shortcuts for common key combos and repetitive operations.

Besides supporting website development, programmers can use the VS Code on desktop applications. Front-end developers use Visual Studio Code's integrated development environment to make their code more effective and error-free.

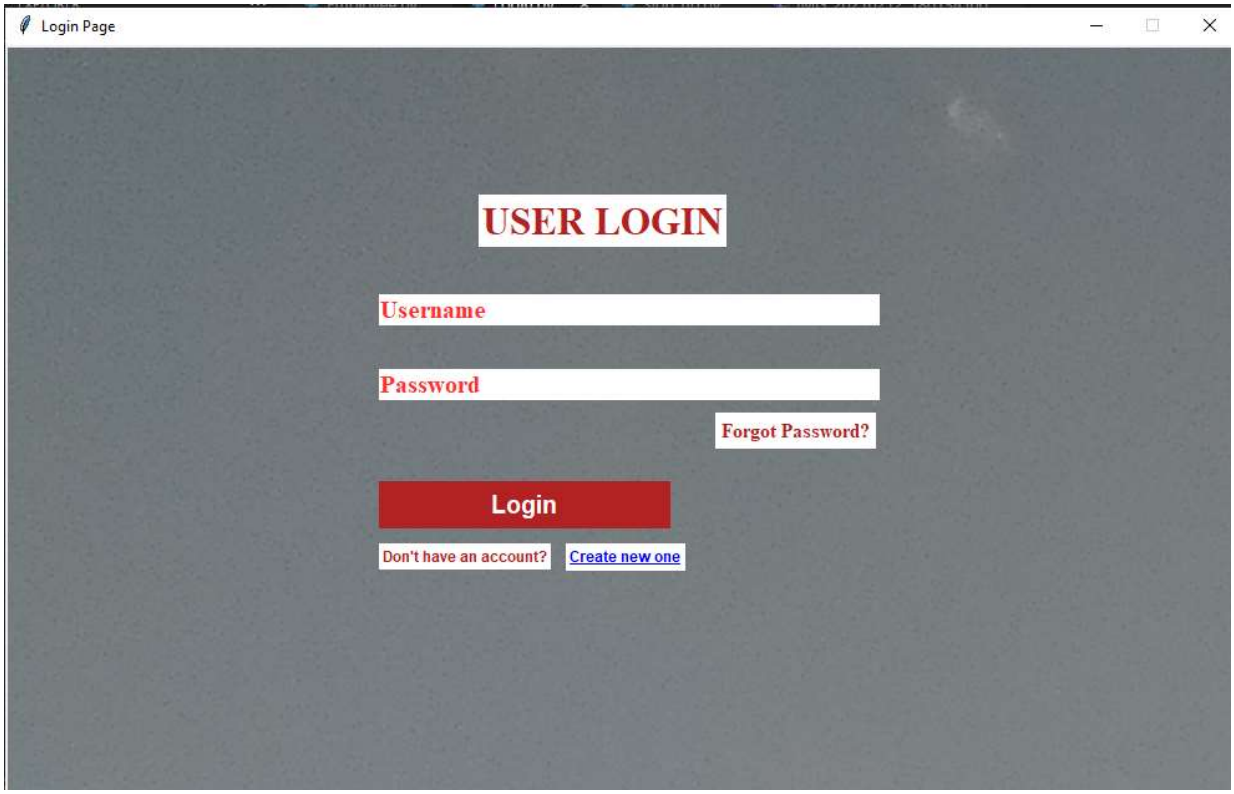
ER-DIAGRAM


USER LOGIN
EMAIL
USERNAME
CREATE NEW ONE
PASSWORD
TERMS & CONDITIONS
CONFIRM PASSWORD
USERNAME
FORGOT PASSWORD
NEW PASSWORD
CONFIRM PASSWORD
LOGIN
EMP_CODE
DESIGNATION
NAME
EMPLOYEE DETAILS
DOB
AGE
DOJ
GENDER
ADDRESS
EMAIL
EXPERIENECE
HIRED LOCATION
CONTACT
STATUS
PROOF_ID
MONTH
YEAR
SALARY
EMPLOYEE SALARY DETAILS
PF
NET SALARY
CONVENCE
MEDICAL
ABSENT
TOTAL DAYS

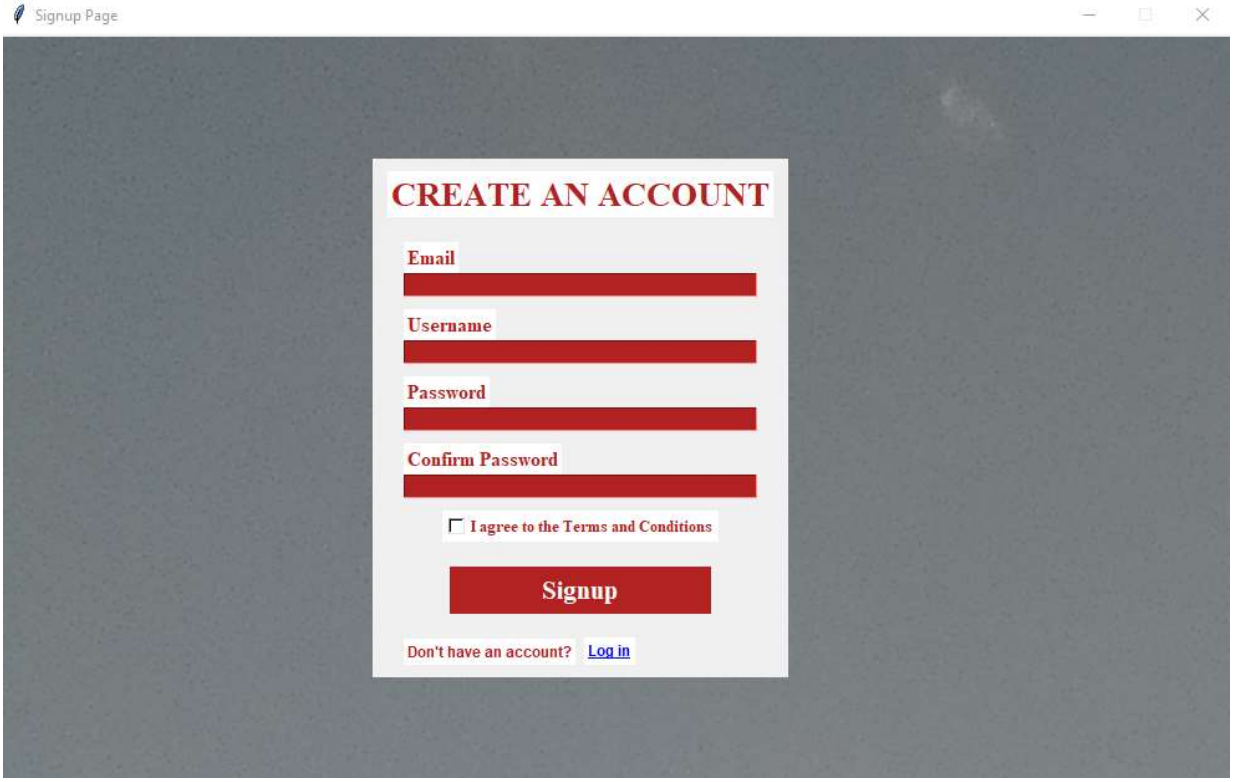
DATA FLOW DIAGRAM (DFD)



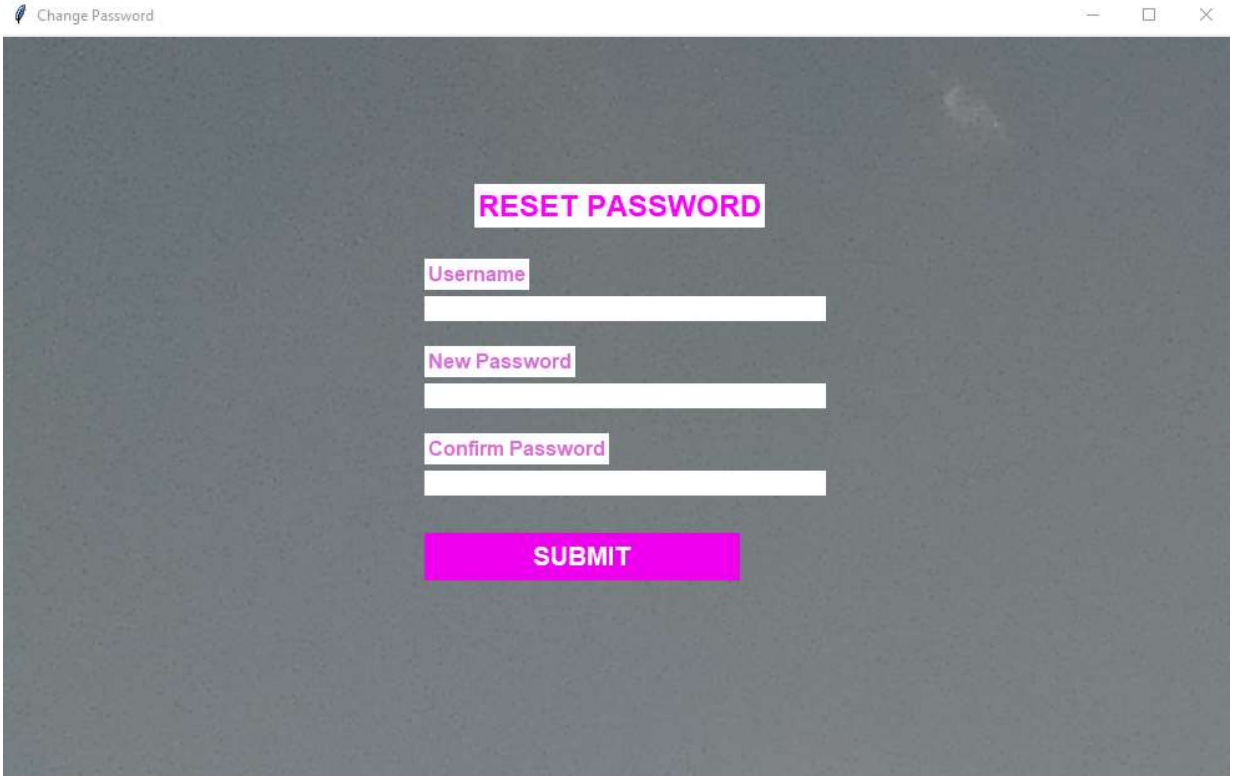
LOGIN FORM


Login Page
USER LOGIN
Username
Password
Forgot Password?
Login
Don't have an account?
Create new one

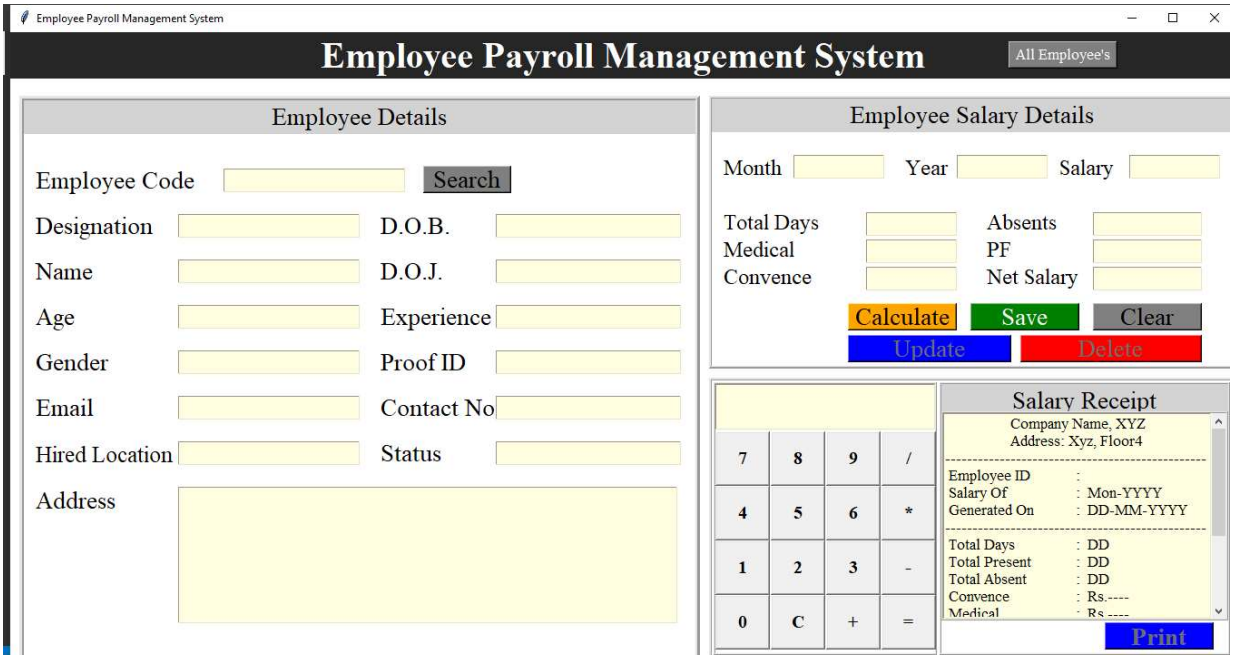
CREATE NEW ACCOUNT FORM



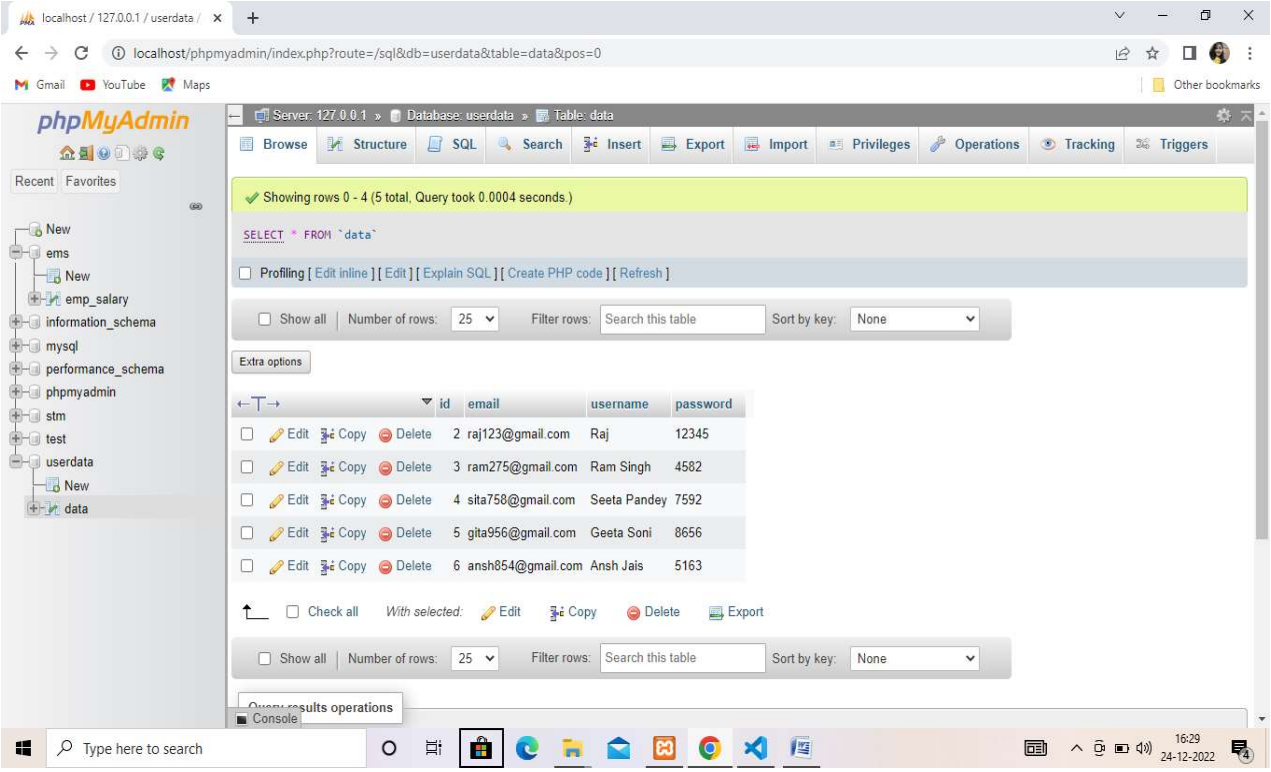
PASSWORD RESET FORM



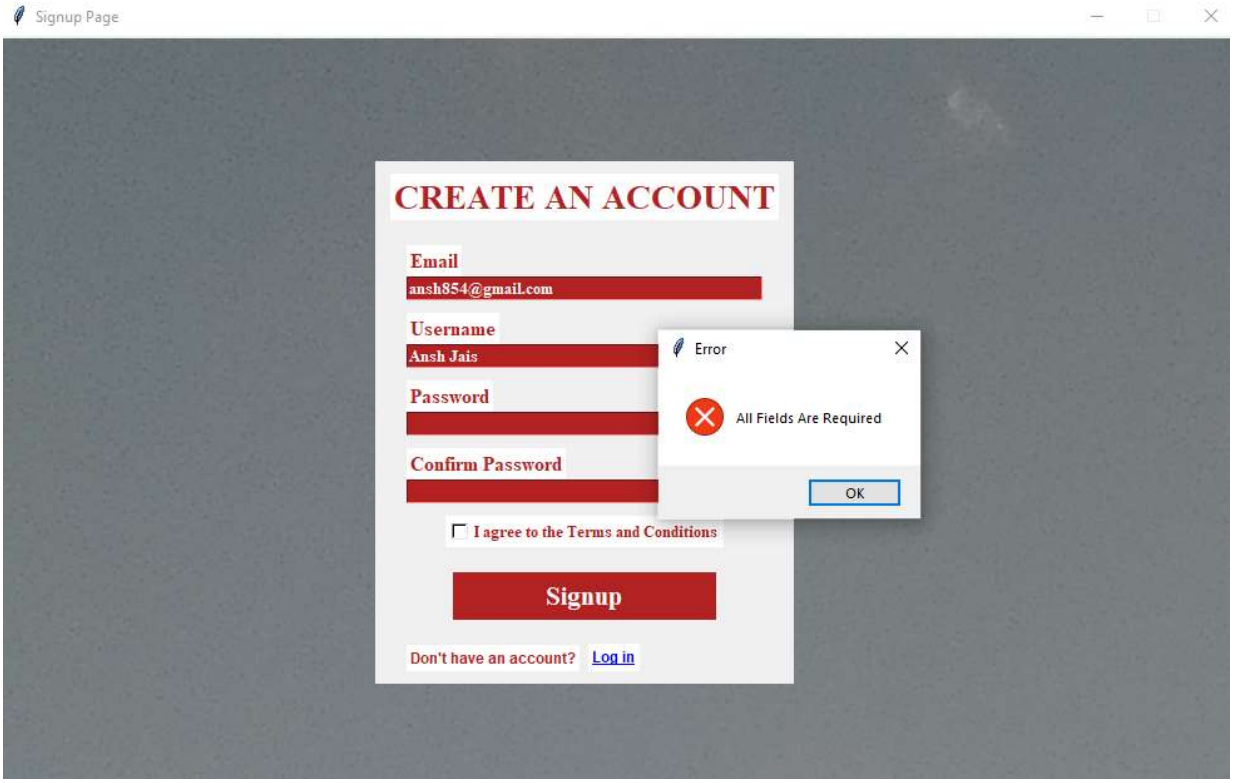
EMPLOYEE DETAILS FORM

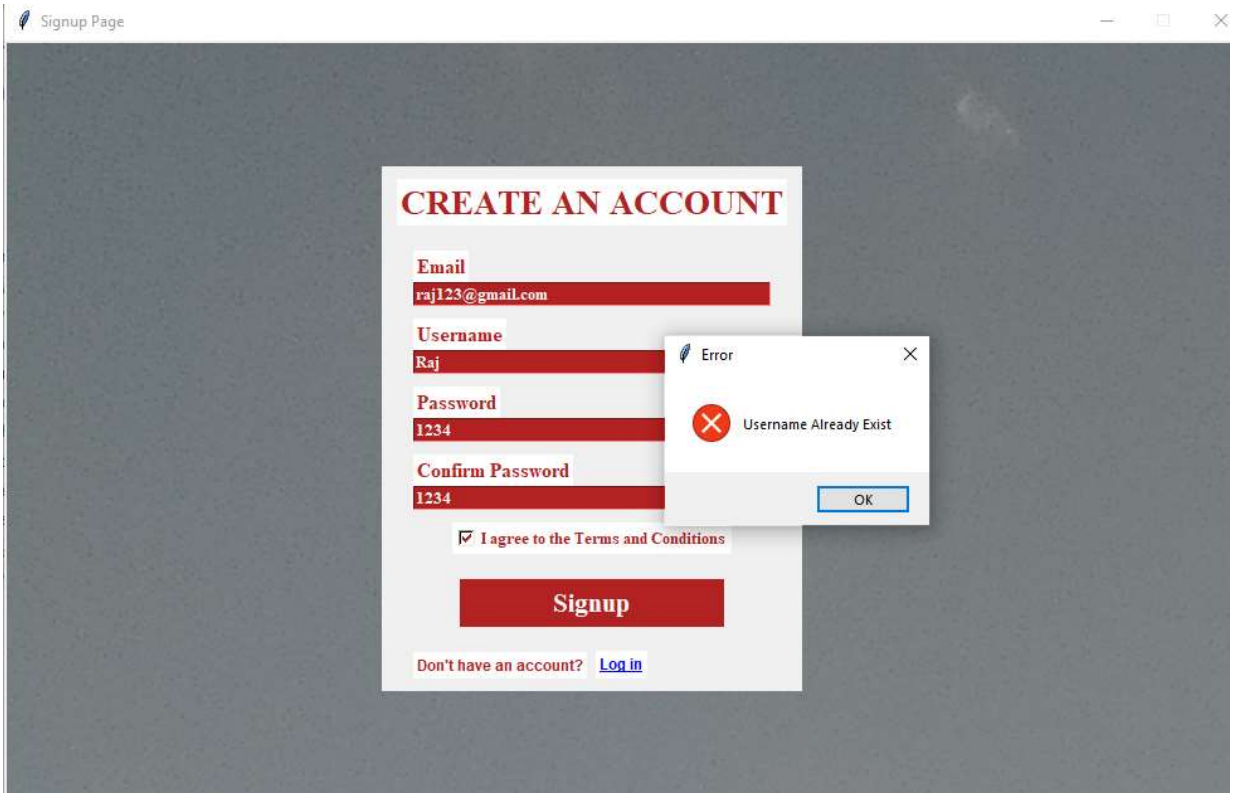


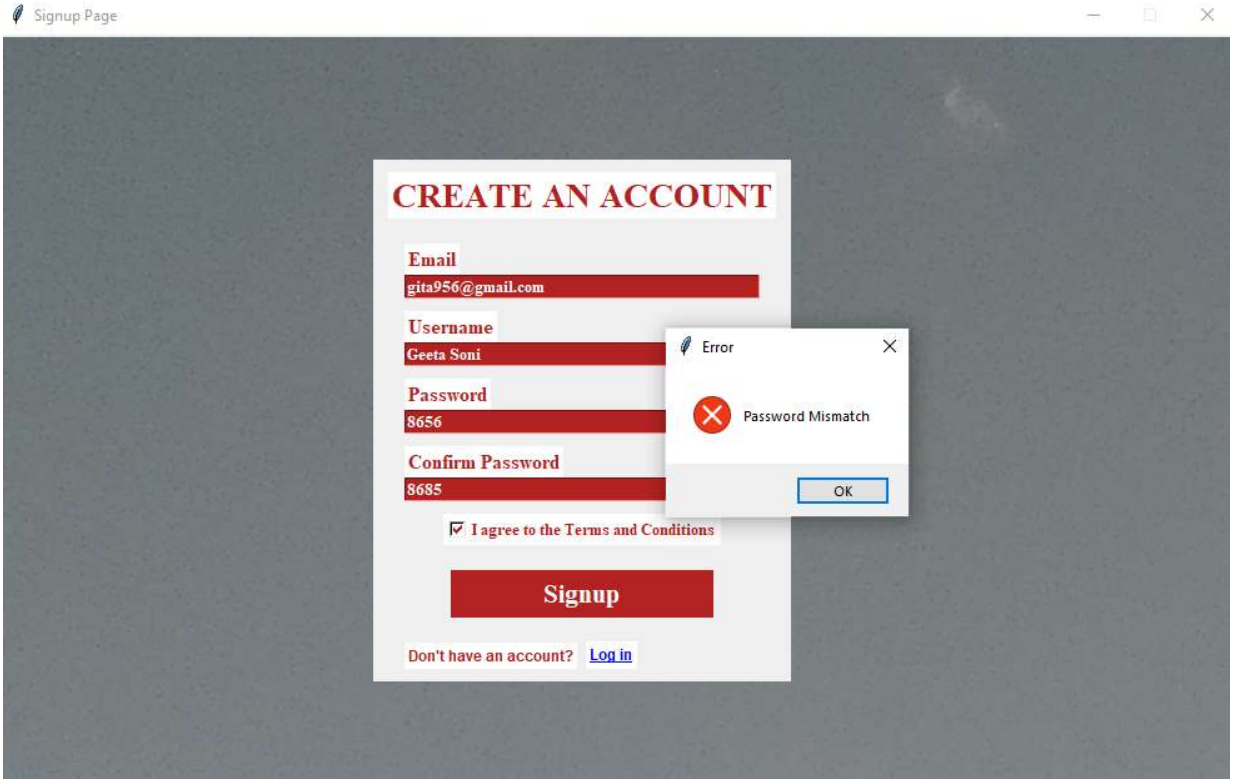
LOGIN FORM DATABASE

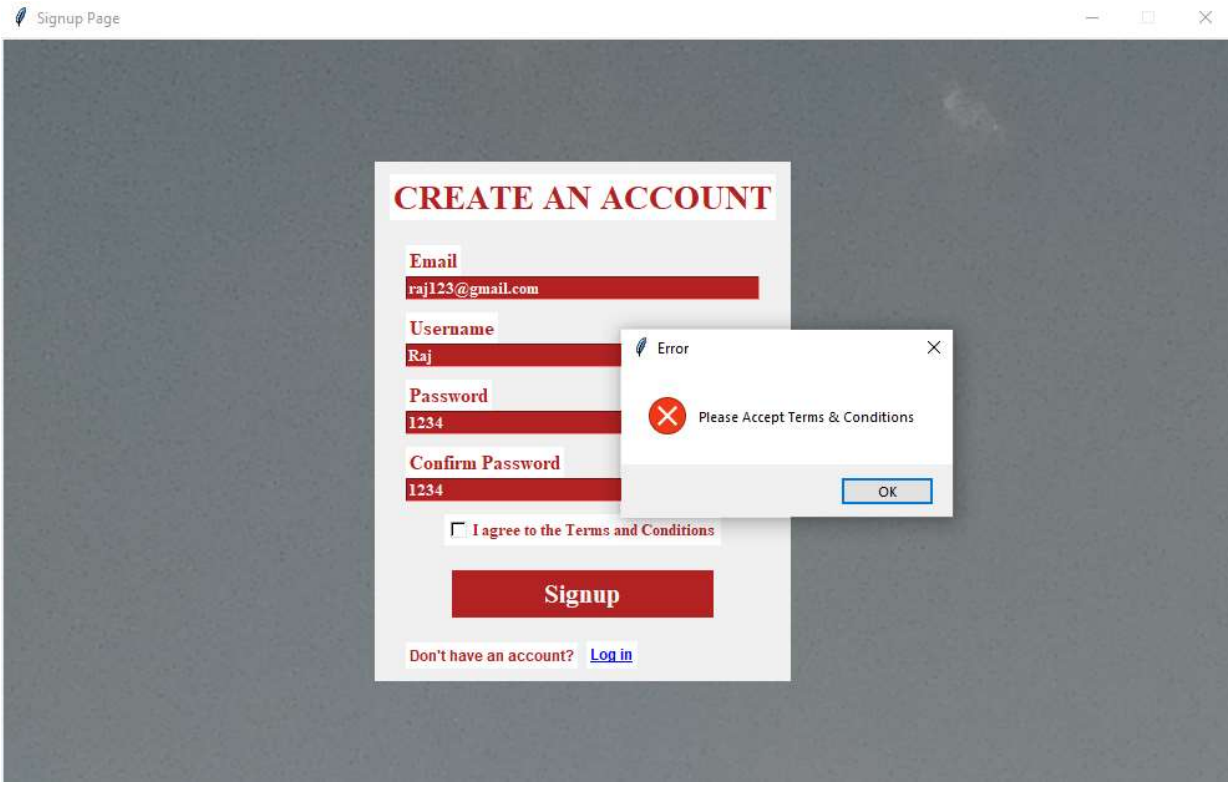


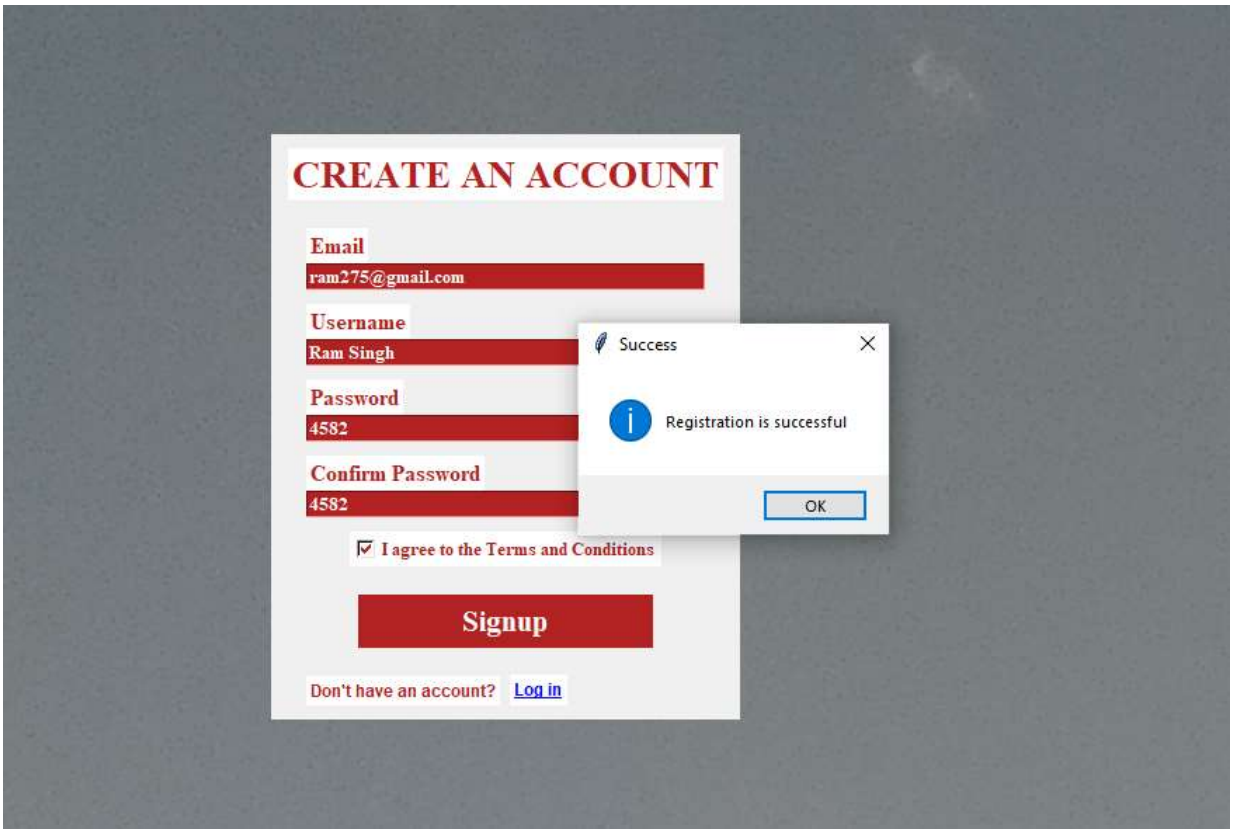
CONDITIONS WHEN WE CREATE NEW ACCOUNT



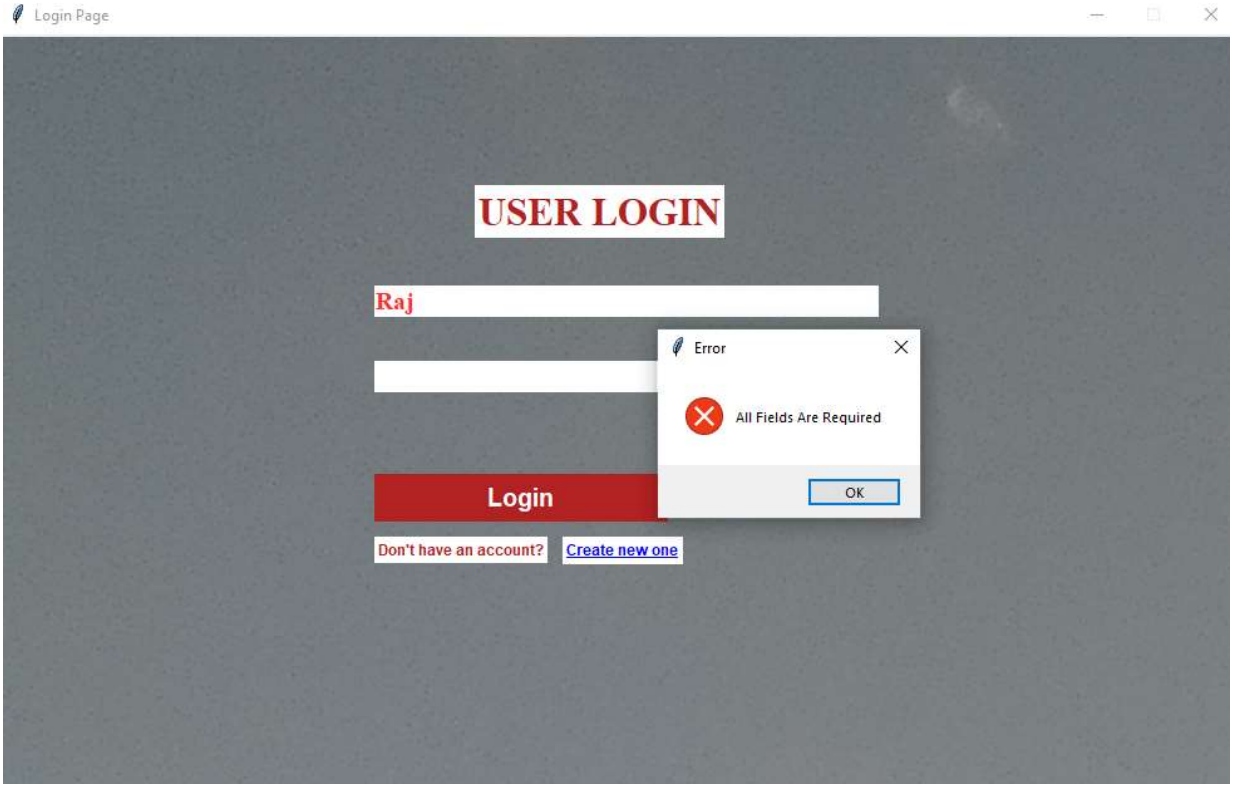

Signup Page
CREATE AN ACCOUNT
Email
raj123@gmail.com
Username
Raj
Password
1234
Confirm Password
1234
I agree to the Terms and Conditions
Signup
Don't have an account?
Log in
Error
Username Already Exist
OK

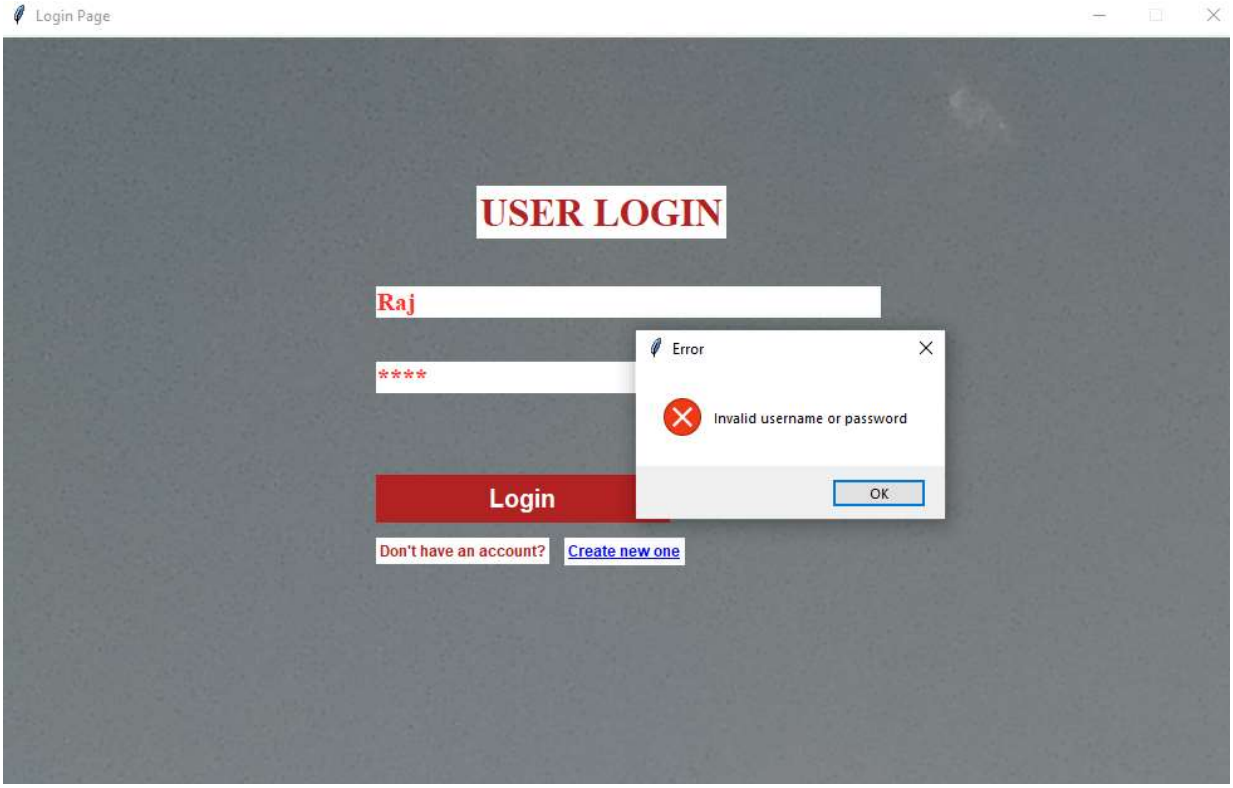

Signup Page
CREATE AN ACCOUNT
Email
gita956@gmail.com
Username
Geeta Soni
Password
8656
Confirm Password
8685
I agree to the Terms and Conditions
Signup
Don't have an account?
Log in
Error
Password Mismatch
OK

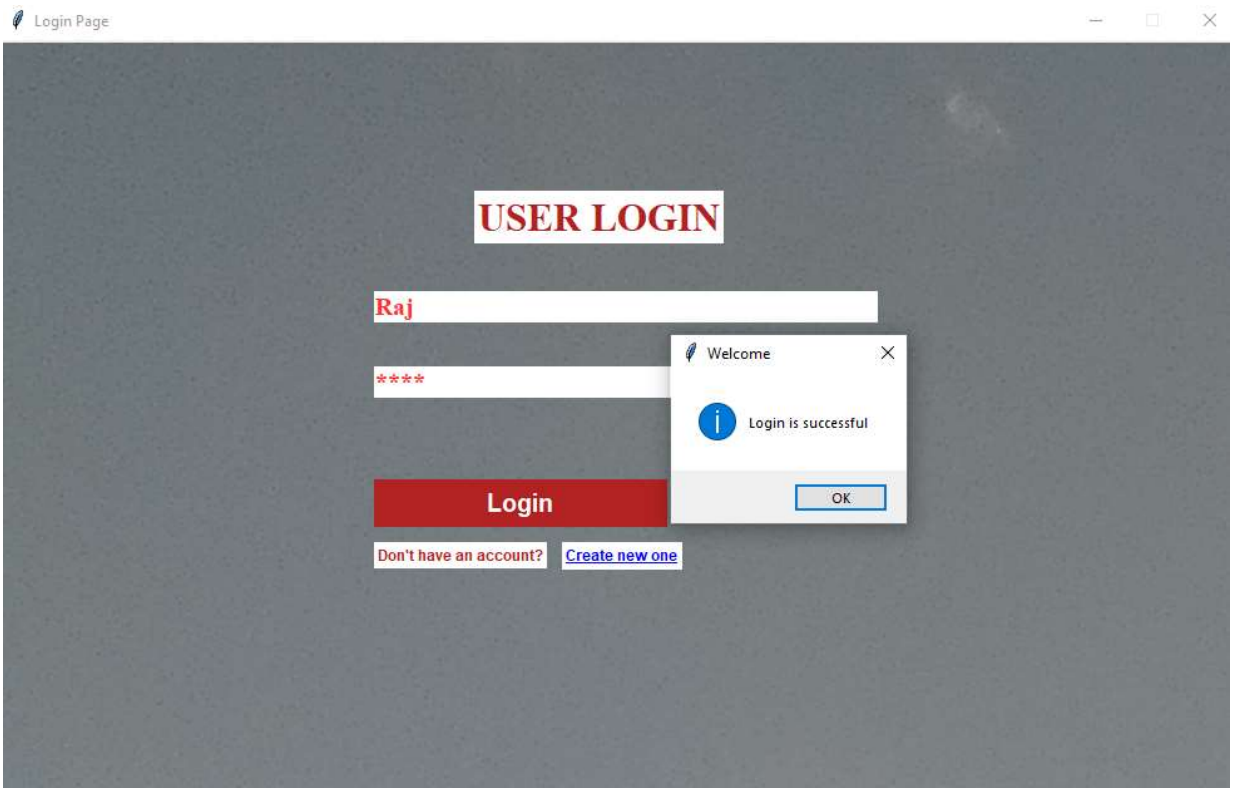

Signup Page
CREATE AN ACCOUNT
Email
raj123@gmail.com
Username
Raj
Password
1234
Confirm Password
1234
I agree to the Terms and Conditions
Signup
Don't have an account?
Log in
Error
Please Accept Terms & Conditions
OK


CREATE AN ACCOUNT
Email
ram275@gmail.com
Username
Ram Singh
Password
4582
Confirm Password
4582
I agree to the Terms and Conditions
Signup
Don't have an account?
Log in
Success
Registration is successful
OK

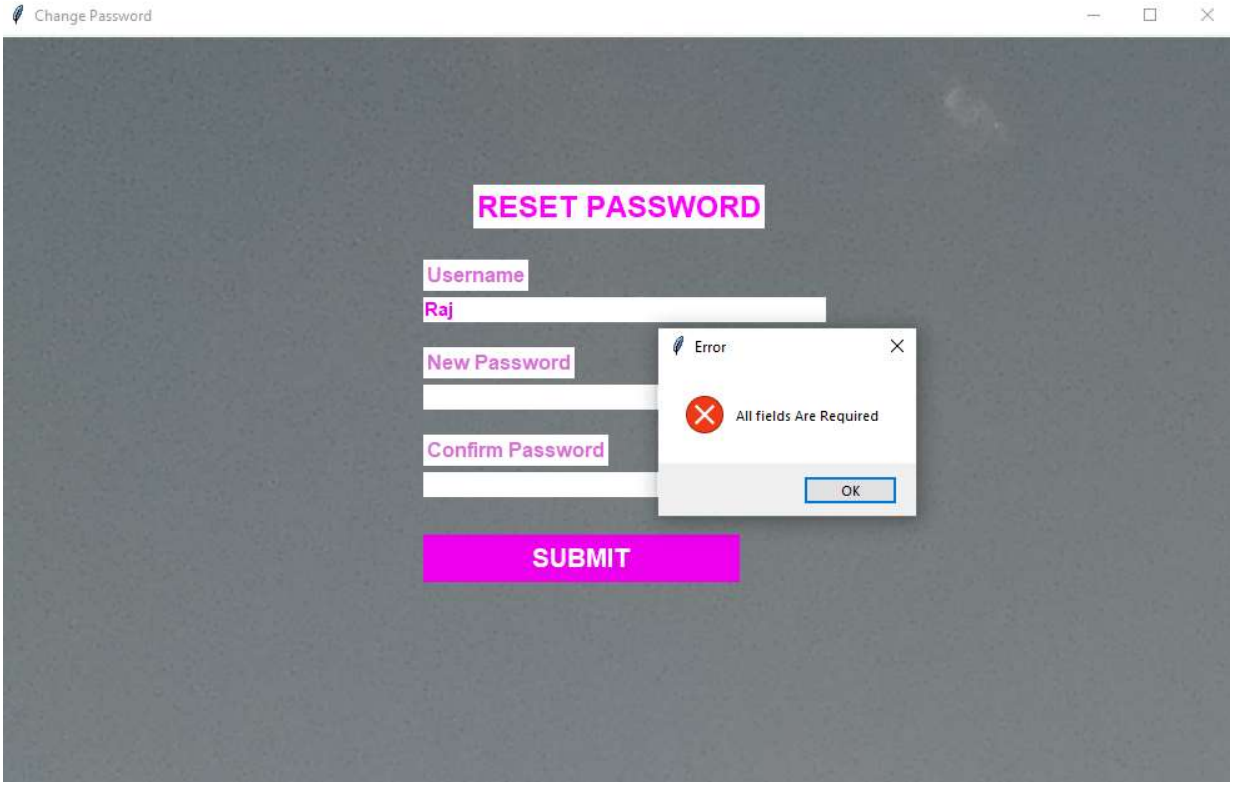
CONDITIONS WHEN WE LOGIN

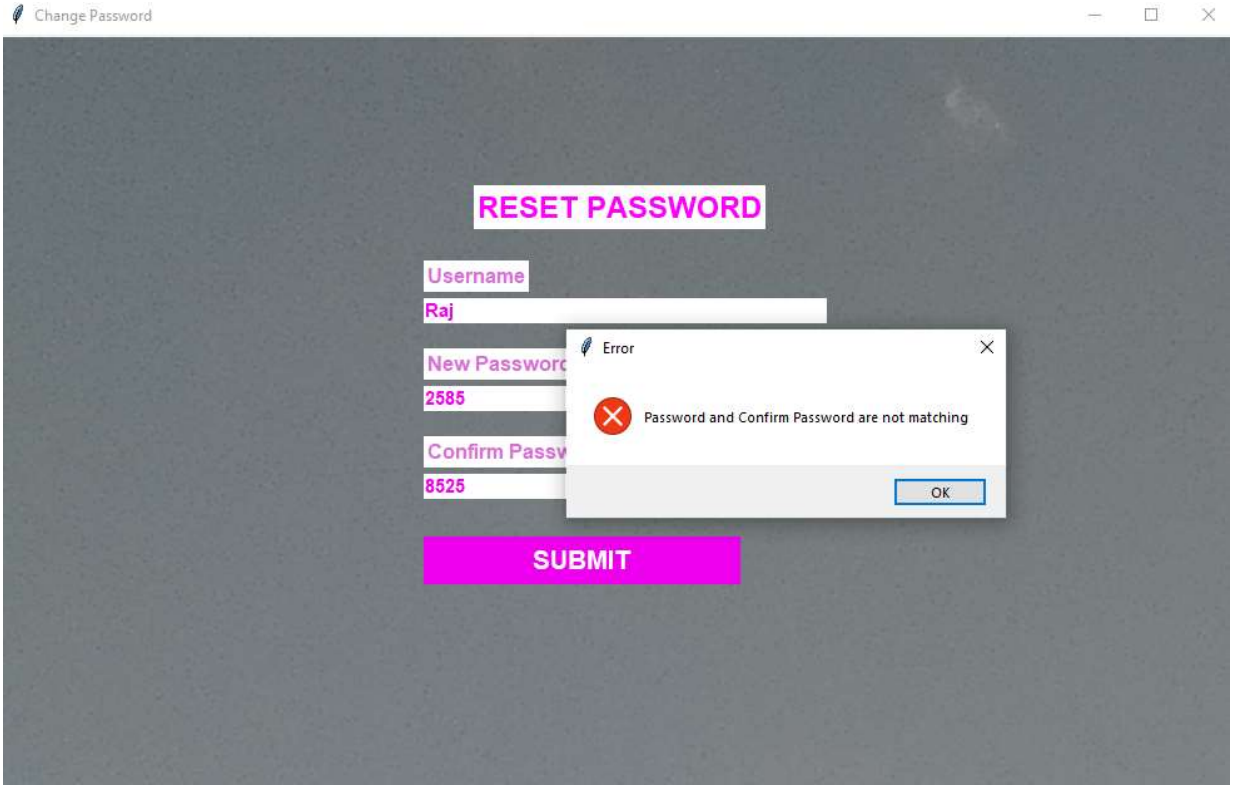


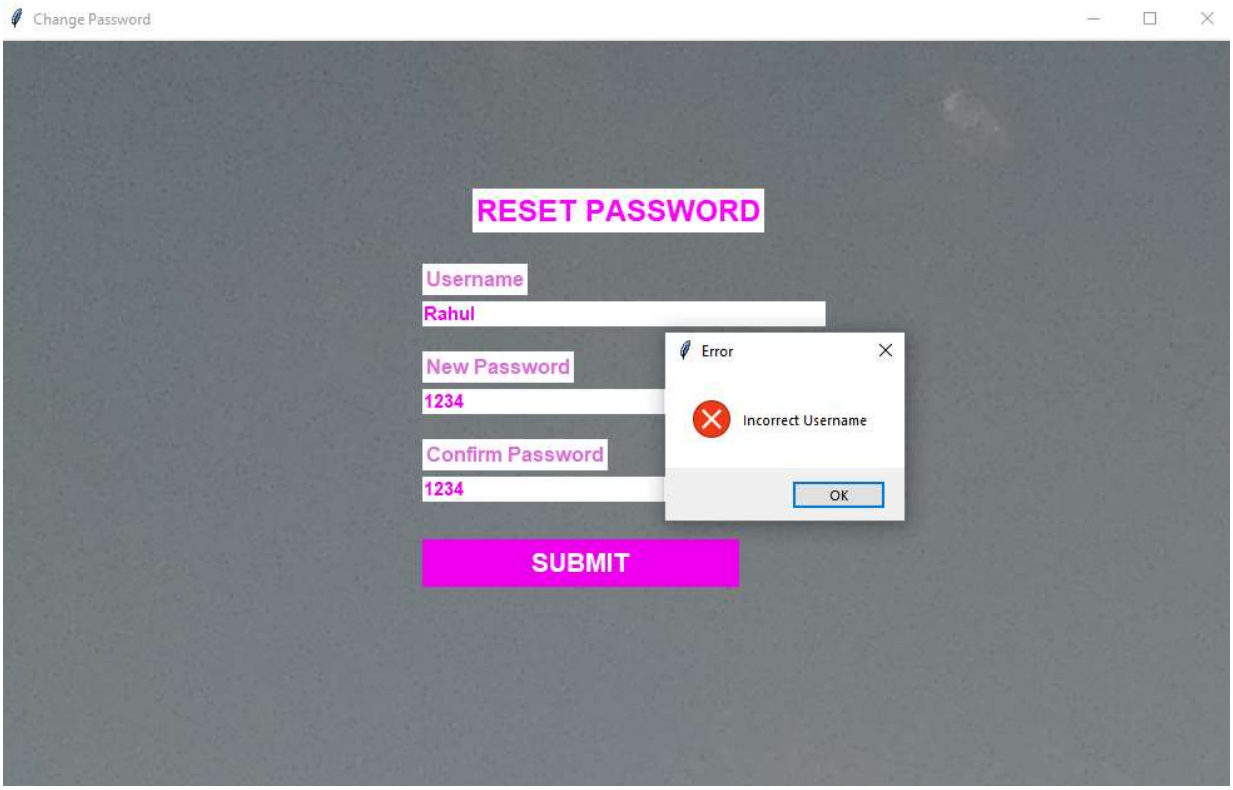


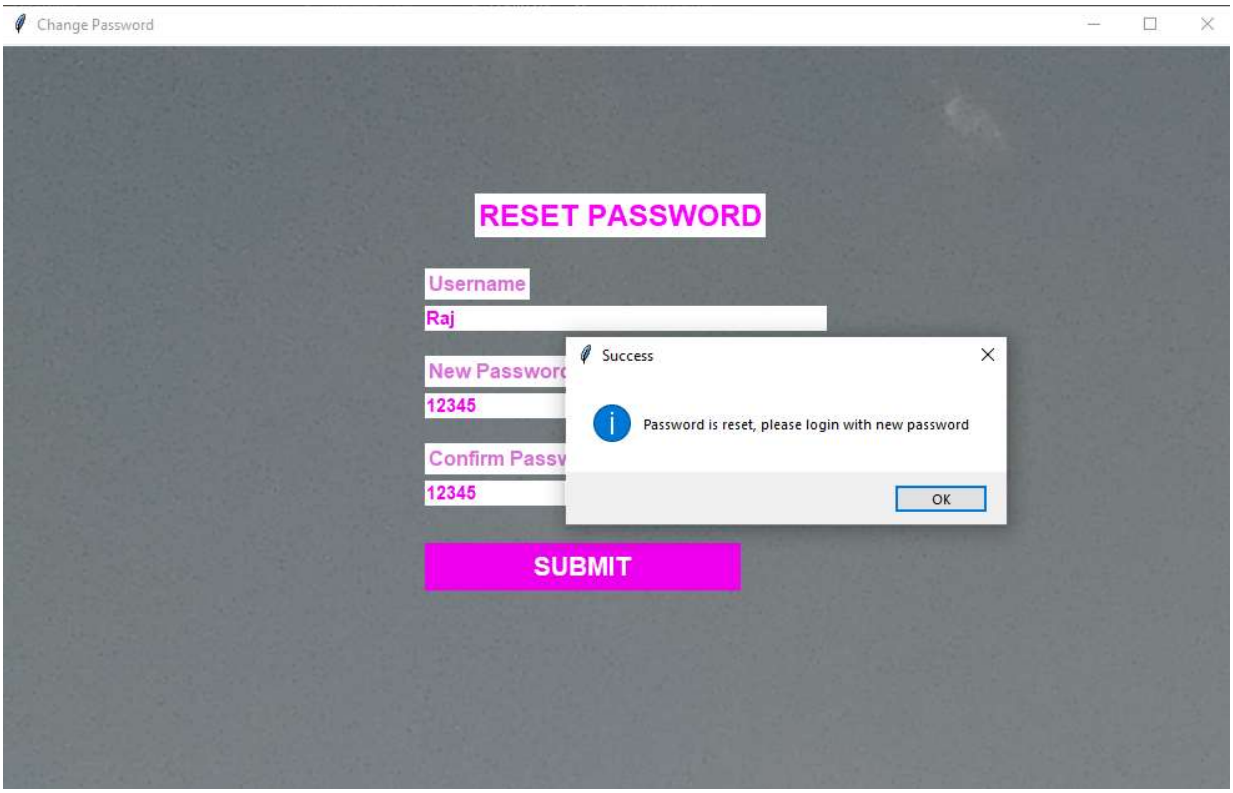

Login Page
USER LOGIN
Raj
****
Welcome
Login is successful
OK
Login
Don't have an account?
Create new one

CONDITIONS WHEN WE FORGET PASSWORD






Change Password
RESET PASSWORD
Username
Rahul
New Password
1234
Confirm Password
1234
SUBMIT
Error
Incorrect Username
OK


Change Password
RESET PASSWORD
Username
Raj
New Passwor
12345
Confirm Passv
12345
SUBMIT
Success
Password is reset, please login with new password
OK

EMPLOYEE DETAILS DATABASE

localhost / 127.0.0.1 / ems / emp

localhost/phpmyadmin/index.php?route=/sql&db=ems&table=emp_salary&pos=0

Server: 127.0.0.1 » Database: ems » Table: emp_salary

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Tracking | Triggers

e_id	designation	name	age	gender	email	hr_location	doj	dob	experience	proof_id	contact	status	address	month	year	basic_salary	total_day
101	Bhopal	Harsh Singh	24	Male	harsh123@gmail.com	Banglore	25/12/2021	12/05/1996	1yr	1523-6525-8965	9214639870	Active	Nagar, Ram Vihar Colony, Bhopal	August	2021	35000	31
102	Jabalpur	Aditya Singh	25	Male	aditya456@gmail.com	Hyderabad	12/08/2020	25/09/1995	2yrs	4521-6352-9874	8569230147	Active	51/8652 Anupam Nagar, Goyar Marg, Jabalpur	September	2022	40000	30
103	Gwalior	Rishita Pathak	23	Female	rishita789@gmail.com	Indore	18/12/2021	25/10/1999	6 months	8523-6914-8569	7548963201	Active	52/4851 Gulab Nagar, Ram Vihar Colony, Gwalior	October	2022	38000	31
104	Indore	Ashika Patel	22	Female	ashika147@gmail.com	Banglore	28/11/2021	12/12/2000	Fresher	1234-5826-4569	9513648790	Work from home	12/5812 Vijay Nagar, Gali No. 2, Indore	May	2022	30000	30

Console

Type here to search 18:10 24-12-2022

localhost / 127.0.0.1 / ems / emp

localhost/phpmyadmin/index.php?route=/sql&db=ems&table=emp_salary&pos=0

Server: 127.0.0.1 » Database: ems » Table: emp_salary

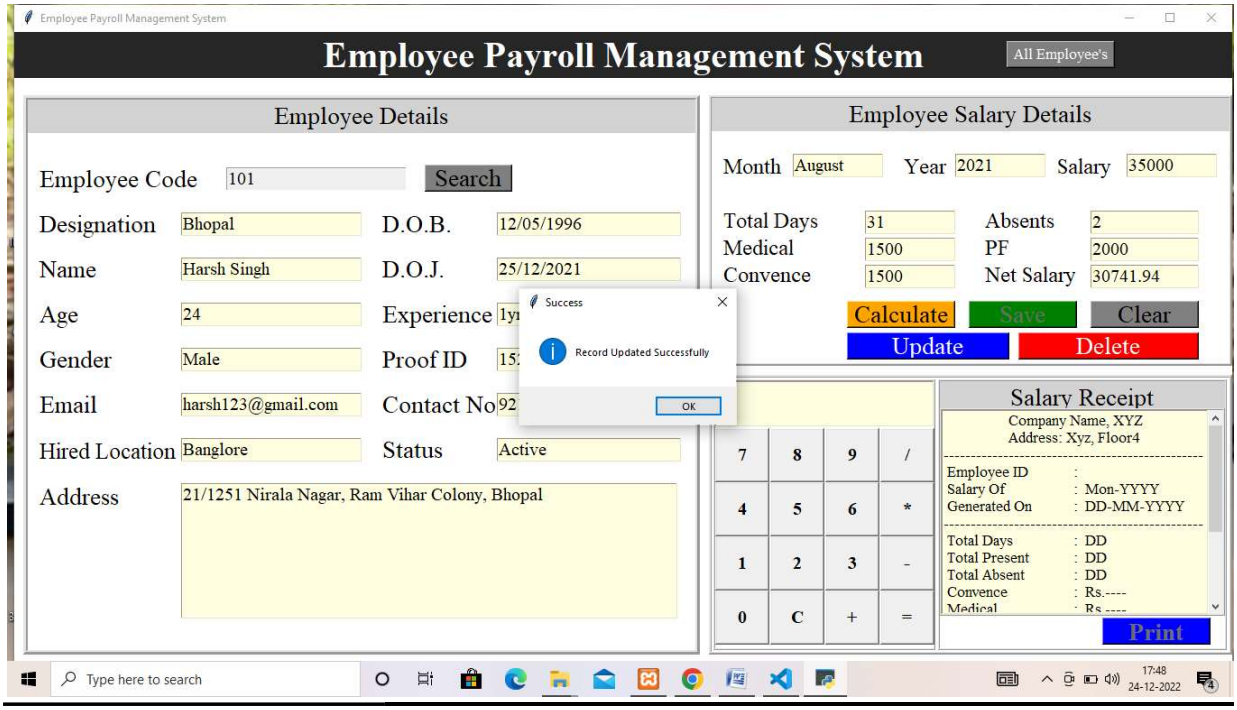
Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Tracking | Triggers

dob	experience	proof_id	contact	status	address	month	year	basic_salary	total_days	absent_days	medical	provident_fund	convence	net_salary	salary_receipt
12/05/1996	1yr	1523-6525-8965	9214639870	Active	Ram Vihar Colony, Bhopal	August	2021	35000	31	2	1500	2000	1500	30741.94	101.txt
25/09/1995	2yrs	4521-6352-9874	8569230147	Active	51/8652 Anupam Nagar, Goyar Marg, Jabalpur	September	2022	40000	30	3	1500	2000	1800	34300.0	102.txt
25/10/1999	6 months	8523-6914-8569	7548963201	Active	52/4851 Gulab Nagar, Ram Vihar Colony, Gwalior	October	2022	38000	31	2	1800	2500	1500	32748.39	103.txt
12/12/2000	Fresher	1234-5826-4569	9513648790	Work from home	12/5812 Vijay Nagar, Gali No. 2, Indore	May	2022	30000	30	1	1500	1500	1500	27500.0	104.txt

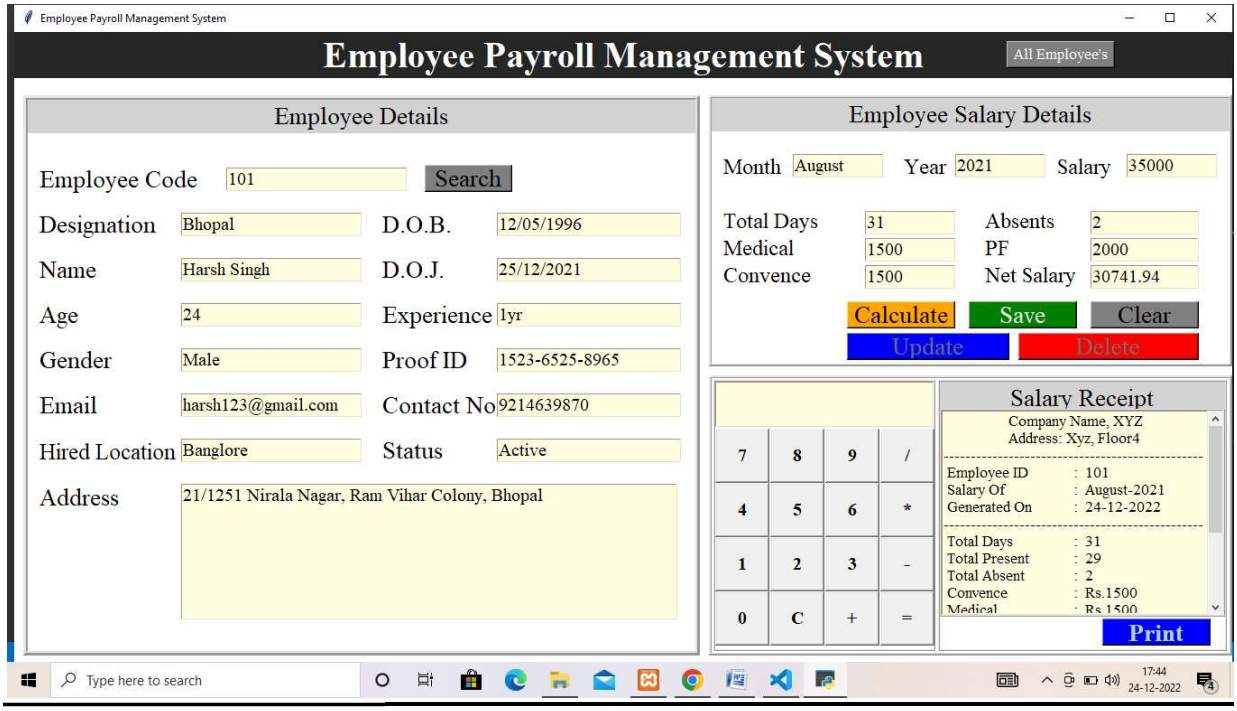
Console

Type here to search 18:10 24-12-2022

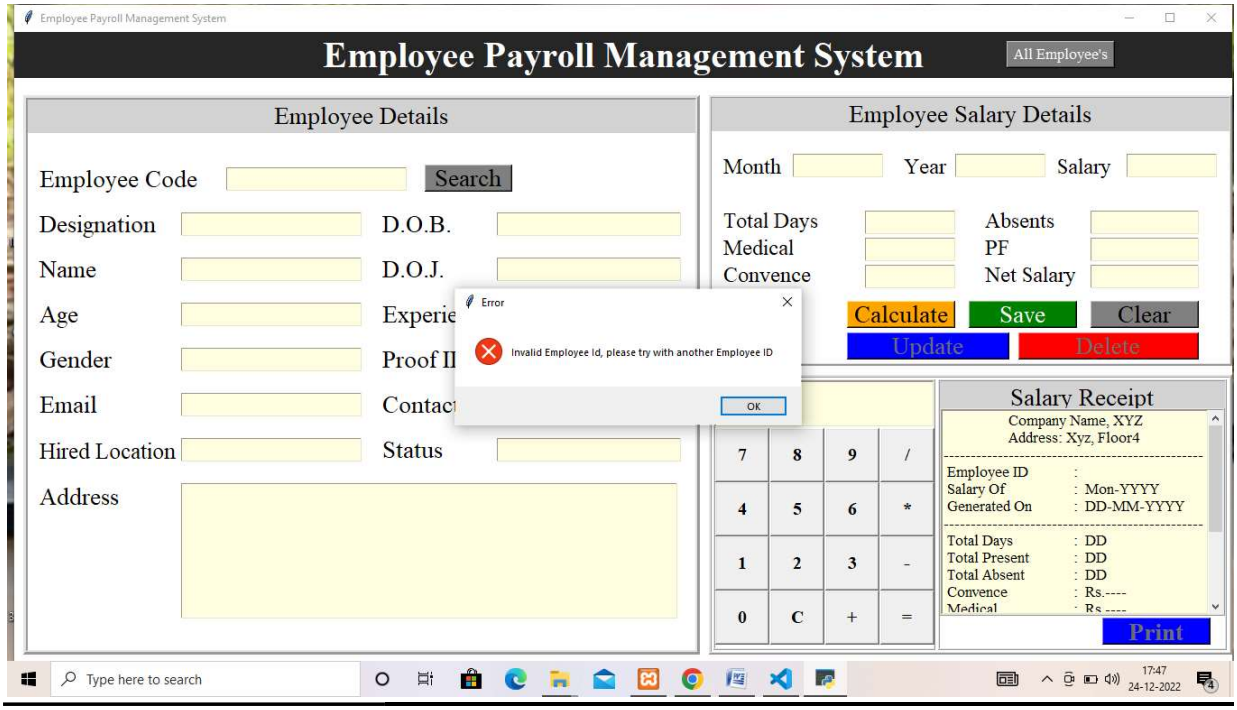
SAVE EMPLOYEE DETAILS



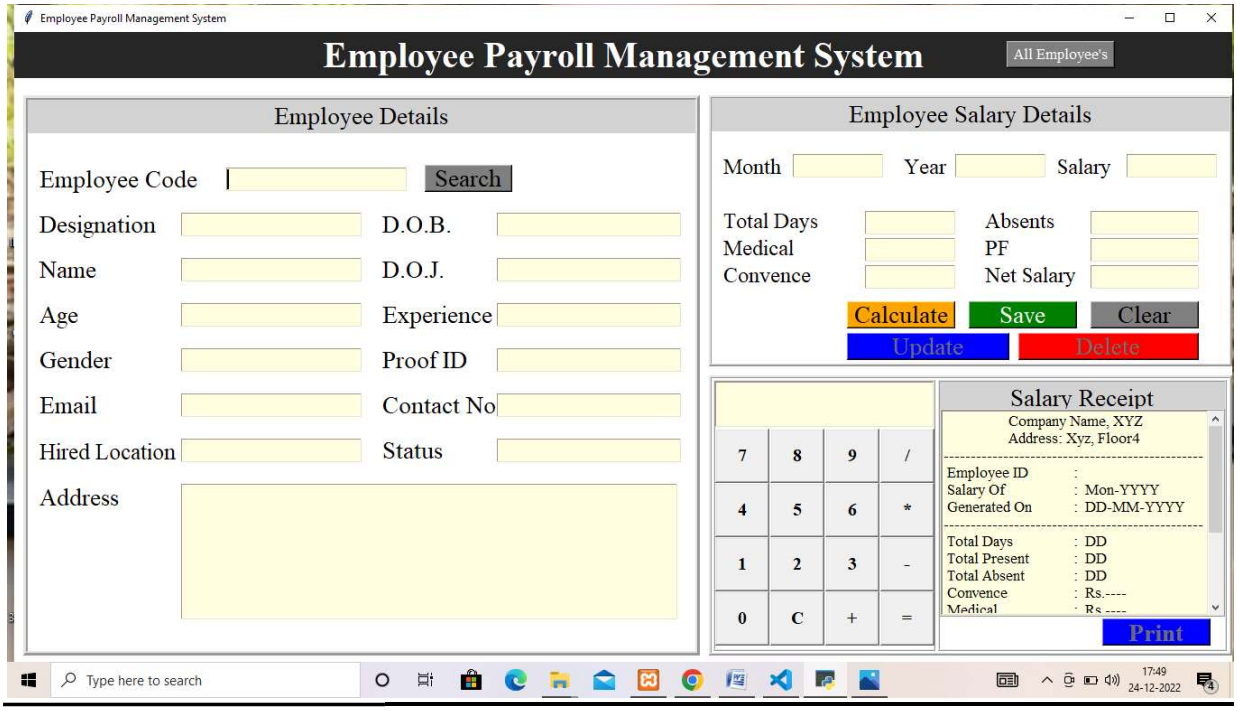
CALCULATE SALARY



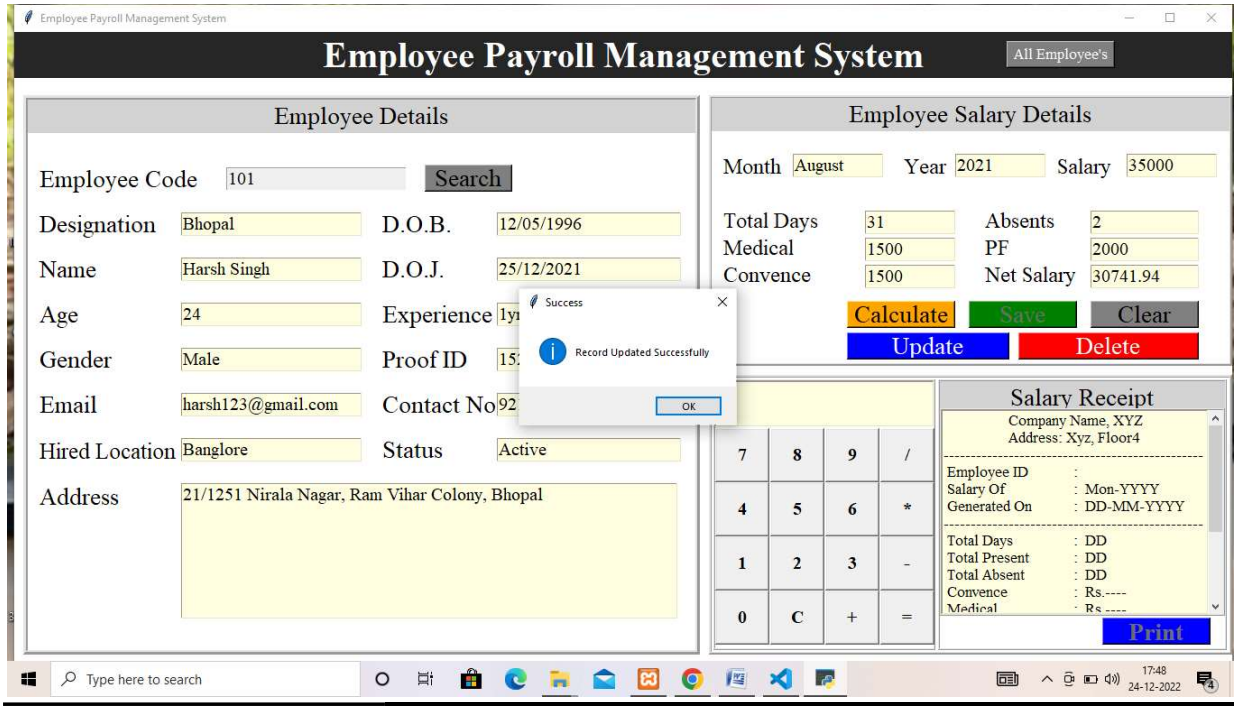
SEARCH EMPLOYEE DETAILS



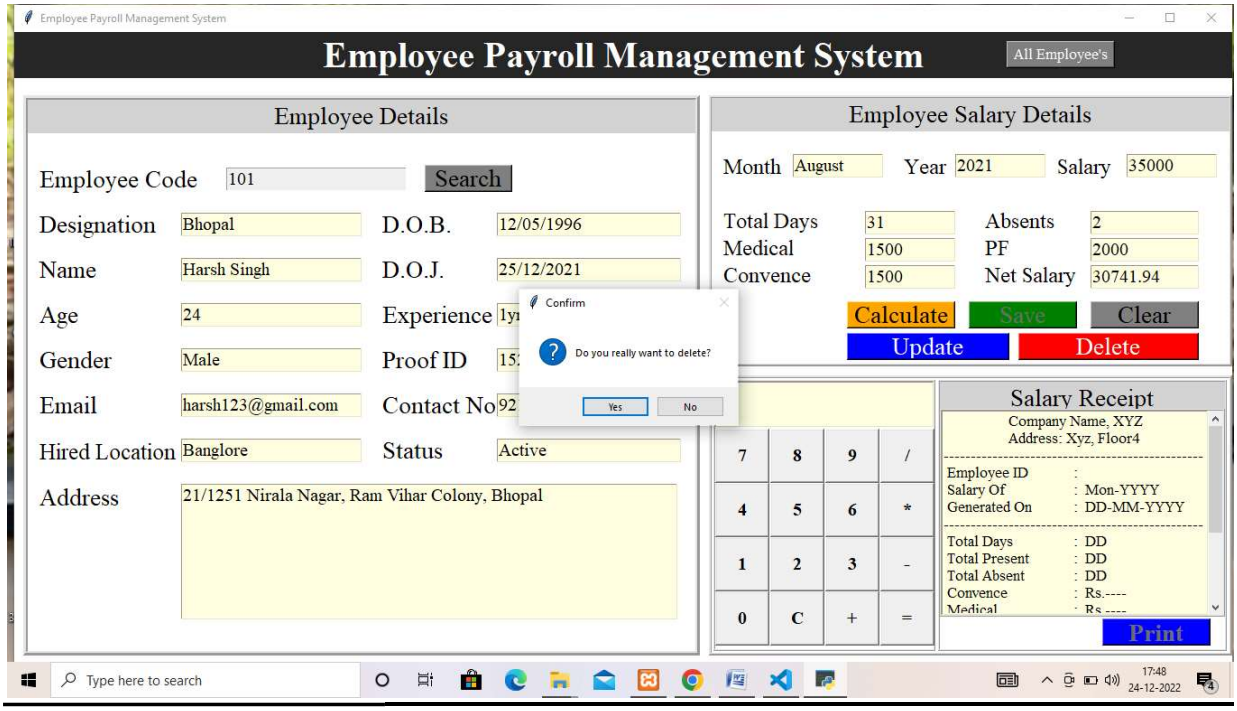
CLEAR EMPLOYEE DETAILS



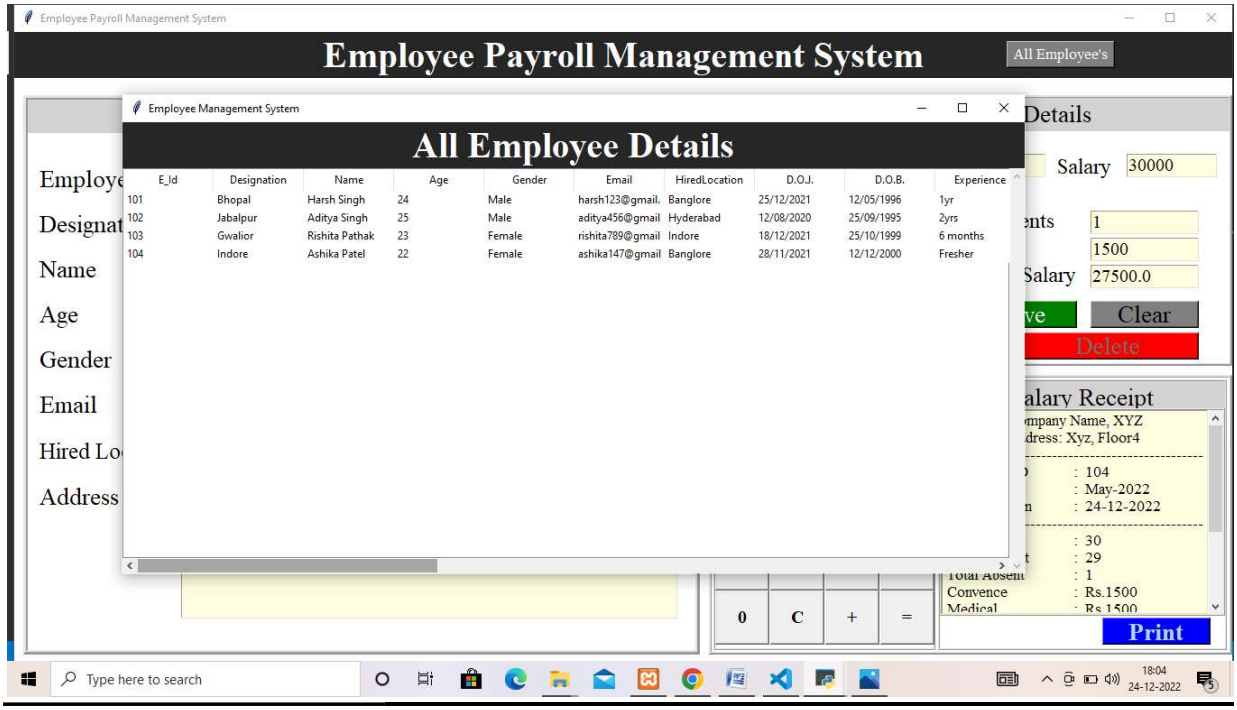
UPDATE EMPLOYEE DETAILS



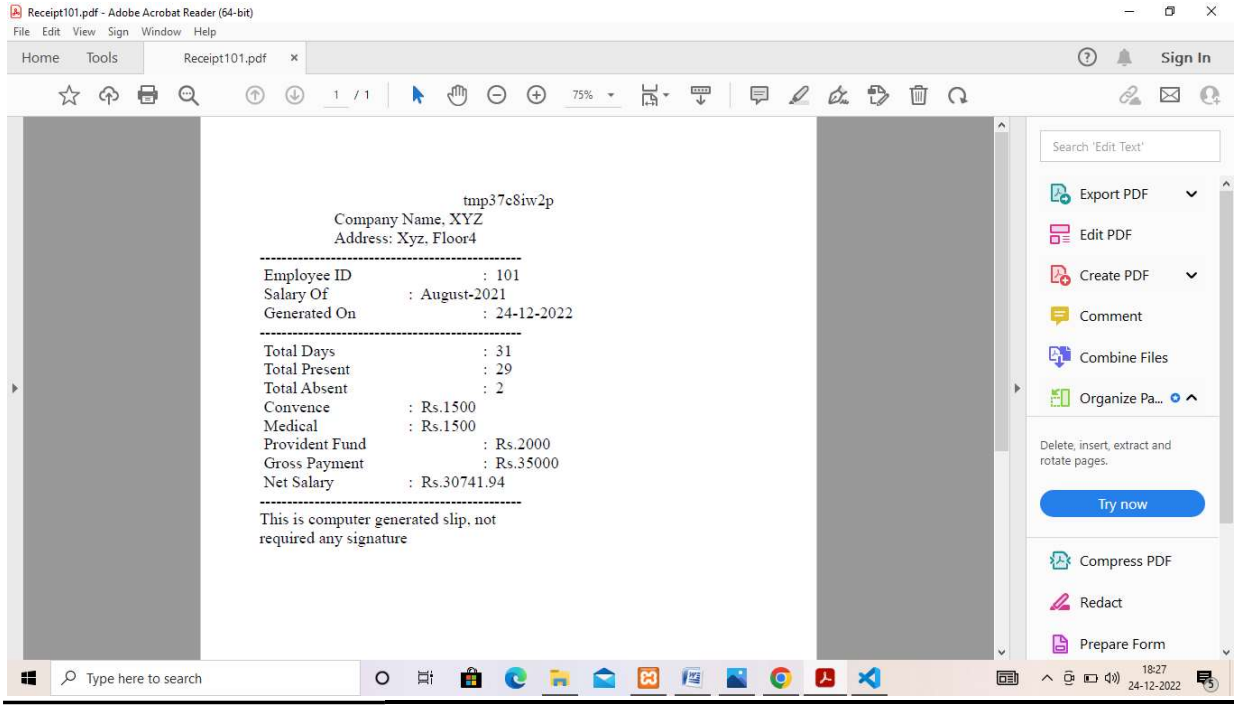
DELETE EMPLOYEE DETAILS



ALL EMPLOYEE DETAILS



EMPLOYEE RECEIPT FORMAT



CODING

Login Form:-

```
from tkinter import*
from tkinter import messagebox
from PIL import ImageTk
import pymysql

#Functionality Part
def  forget_pass():
    def change_password():
        if user_entry.get()=='' or newpass_entry.get()=='' or
confirmpass_entry.get()=='':
            messagebox.showerror('Error','All fields Are Required',parent=window)
        elif newpass_entry.get() != confirmpass_entry.get():
            messagebox.showerror('Error','Password and Confirm Password are not
matching',parent=window)
        else:
con=pymysql.connect(host='localhost',user='root',password='',database='userdata'
)
            mycursor=con.cursor()
            query = 'select * from data where username=%s'
            mycursor.execute(query, (user_entry.get()))
            row=mycursor.fetchone()
            if row==None:
                messagebox.showerror('Error','Incorrect Username',parent=window)
            else:
                query='update data set password=%s where username=%s'
                mycursor.execute(query,(newpass_entry.get(),user_entry.get()))
                con.commit()
                con.close()
                messagebox.showinfo('Success','Password is reset, please login with
new password',parent=window)
                window.destroy()
    window = Toplevel()
    window.title('Change Password')
    window.geometry('990x600+50+50')
```

```
    bgPic = ImageTk.PhotoImage(file='IMG_20210212_180134.jpg')
    bglabel = Label(window, image=bgPic)
    bglabel.grid()
    heading_label = Label(window, text='RESET
PASSWORD',font=('arial','18','bold'),
    bg='white',fg='magenta')
    heading_label.place(x=380,y=120)

    userlabel = Label(window, text='Username',
font=('arial',12,'bold'),bg='white',fg='orchid')userlabel.place(x=340,y=180)
    user_entry = Entry(window, width=40, fg='magenta2',
font=('arial',11,'bold'),bd=0)user_entry.place(x=340,y=210)

    passwordlabel = Label(window, text='New Password',
font=('arial',12,'bold'),bg='white',fg='orchid')passwordlabel.place(x=340,y=250)
    newpass_entry = Entry(window, width=40, fg='magenta2',
font=('arial',11,'bold'),bd=0)newpass_entry.place(x=340,y=280)

    confirmpasslabel = Label(window, text='Confirm Password',
font=('arial',12,'bold'),bg='white',fg='orchid')
    confirmpasslabel.place(x=340,y=320)
    confirmpass_entry = Entry(window, width=40, fg='magenta2',
font=('arial',11,'bold'),bd=0)confirmpass_entry.place(x=340,y=350)

    submitButton = Button(window,
text='SUBMIT',bd=0,bg='magenta2',fg='white',font=('Open Sans','16','bold'),

width=19,cursor='hand2',activebackground='magenta2',activeforeground='white',
command=change_password)
    submitButton.place(x=340,y=400)
    window.mainloop()

def signup_page():
    login_window.destroy()
    import Sign_up

def Login():
```

```
    if usernameEntry.get()==" or passwordEntry.get()==":
        messagebox.showerror('Error','All Fields Are Required')
    else:
        try:
            con=pymysql.connect(host='localhost',user='root',password='')
            mycursor=con.cursor()
        except:
            messagebox.showerror('Error','Connection is not established try again')
            return
        query = 'use userdata'
        mycursor.execute(query)
        query='select * from data where username=%s and password=%s'
        mycursor.execute(query,(usernameEntry.get(),passwordEntry.get()))
        row=mycursor.fetchone()
        if row==None:
            messagebox.showerror('Error','Invalid username or password')
        else:
            messagebox.showinfo('Welcome','Login is successful')
            login_window.destroy()
            import Employee
def user_enter(event):
    if usernameEntry.get()=='Username':
        usernameEntry.delete(0,END)

def password_enter(event):
    if passwordEntry.get()=='Password':
        passwordEntry.delete(0,END)
        passwordEntry.config(show='*')

#Gui Part
login_window=Tk()
login_window.geometry('990x600+50+50')
login_window.resizable(0,0)
login_window.title('Login Page')
bgImage=ImageTk.PhotoImage(file='IMG_20210212_180134.jpg')

bgLabel=Label(login_window,image=bgImage)
```

```
bgLabel.place(x=0,y=0)
heading=Label(login_window,text='USER LOGIN',font=('Times New
Roman',23,'bold'),bg='white',fg='firebrick')
heading.place(x=380,y=120)

usernameEntry=Entry(login_window,width=40,font=('Times New
Roman',15,'bold'),bd=0,fg='firebrick1')
usernameEntry.place(x=300,y=200)
usernameEntry.insert(0,'Username')
usernameEntry.bind('<FocusIn>',user_enter)
passwordEntry=Entry(login_window,width=40,font=('Times New
Roman',15,'bold'),bd=0,fg='firebrick1')
passwordEntry.place(x=300,y=260)
passwordEntry.insert(0,'Password')
passwordEntry.bind('<FocusIn>',password_enter)

forgetButton=Button(login_window,text='Forgot
Password?',bd=0,bg='white',activebackground='white',cursor='hand2'
,font=('Times New Roman',12,'bold'),fg='firebrick',command=forget_pass)
forgetButton.place(x=570,y=295)

loginButton=Button(login_window,text='Login',font=('Open
Sans',15,'bold'),fg='white',bg='firebrick'
,activeforeground='white',activebackground='firebrick',cursor='hand2',bd=0,widt
h=19,command=Login)
loginButton.place(x=300,y=350)
signupLabel=Label(login_window,text="Don't have an account?",font=('Open
Sans',9,'bold'),fg='firebrick',bg='white')
signupLabel.place(x=300,y=400)

newaccountButton=Button(login_window,text='Create new one',font=('Open
Sans',9,'bold underline'),fg='blue',bg='white'
,activeforeground='blue',activebackground='firebrick',cursor='hand2',bd=0,comm
and=signup_page)
newaccountButton.place(x=450,y=400)

login_window.mainloop()
```

Signup Form:-

```
from tkinter import *
from tkinter import messagebox
from PIL import ImageTk
import pymysql

def clear():
    emailEntry.delete(0,END)
    usernameEntry.delete(0,END)
    passwordEntry.delete(0,END)
    confirmEntry.delete(0,END)
    check.set(0)

def connect_database():
    if emailEntry.get()== '' or usernameEntry.get()== '' or passwordEntry.get()== ''
or confirmEntry.get()== '':
        messagebox.showerror('Error','All Fields Are Required')
    elif passwordEntry.get() != confirmEntry.get():
        messagebox.showerror('Error','Password Mismatch')
    elif check.get()==0:
        messagebox.showerror('Error','Please Accept Terms & Conditions')
    else:
        try:
            con=pymysql.connect(host='localhost',user='root',password='')
            mycursor=con.cursor()
        except:
            messagebox.showerror('Error','Database Connectvity Issue, Please Try
Again')
            return
        try:
            query='create database userdata'
            mycursor.execute(query)
            query='use userdata'
            mycursor.execute(query)
```

```
        query='create table data(id int auto_increment primary key not null,
email varchar(50),username varchar(100),password varchar(20))'
        mycursor.execute(query)
    except:
        mycursor.execute('use userdata')

    query='select * from data where username=%s'
    mycursor.execute(query,(usernameEntry.get()))

    row=mycursor.fetchone()
    if row != None:
        messagebox.showerror('Error','Username Already Exist')

    else:
        query='insert into data(email,username,password) values(%s,%s,%s)'

mycursor.execute(query,(emailEntry.get(),usernameEntry.get(),passwordEntry.g
et()))
        con.commit()
        con.close()
        messagebox.showinfo('Success','Registration is successful')
        clear()
        signup_window.destroy()
        import Login


def Login():
    signup_window.destroy()
    import Login

signup_window=Tk()
signup_window.geometry('990x600+50+50')
signup_window.title("Signup Page")
signup_window.resizable(False,False)
background=ImageTk.PhotoImage(file='IMG_20210212_180134.jpg')
```

```
bgLabel=Label(signup_window,image=background)
bgLabel.grid()

frame=Frame(signup_window)
frame.place(x=300,y=100)

heading=Label(frame,text='CREATE AN ACCOUNT',font=('Times New
Roman',20,'bold'),bg='white',fg='firebrick')
heading.grid(row=0,column=0,padx=10,pady=10)

emailLabel=Label(frame,text='Email',font=('Times New
Roman',12,'bold'),bg='white',fg='firebrick')
emailLabel.grid(row=2,column=0,sticky='w',padx=25,pady=(10,0))

emailEntry=Entry(frame,width=40,font=('Times New
Roman',10,'bold'),fg='white',bg='firebrick')
emailEntry.grid(row=3,column=0,sticky='w',padx=25)

usernameLabel=Label(frame,text='Username',font=('Times New
Roman',12,'bold'),bg='white',fg='firebrick')
usernameLabel.grid(row=4,column=0,sticky='w',padx=25,pady=(10,0))

usernameEntry=Entry(frame,width=40,font=('Times New
Roman',10,'bold'),fg='white',bg='firebrick')
usernameEntry.grid(row=5,column=0,sticky='w',padx=25)

passwordLabel=Label(frame,text='Password',font=('Times New
Roman',12,'bold'),bg='white',fg='firebrick')
passwordLabel.grid(row=6,column=0,sticky='w',padx=25,pady=(10,0))

passwordEntry=Entry(frame,width=40,font=('Times New
Roman',10,'bold'),fg='white',bg='firebrick')
passwordEntry.grid(row=7,column=0,sticky='w',padx=25)
```

```
confirmLabel=Label(frame,text='Confirm Password',font=('Times New
Roman',12,'bold'),bg='white',fg='firebrick')
confirmLabel.grid(row=8,column=0,sticky='w',padx=25,pady=(10,0))

confirmEntry=Entry(frame,width=40,font=('Times New
Roman',10,'bold'),fg='white',bg='firebrick')
confirmEntry.grid(row=9,column=0,sticky='w',padx=25)
check=IntVar()

termsandconditions=Checkbutton(frame,text='I agree to the Terms and
Conditions',font=('Times New Roman',10,'bold'),
fg='firebrick',bg='white',activebackground='white',activeforeground='firebrick',c
ursor='hand2',variable=check)
termsandconditions.grid(row=10,column=0,padx=15,pady=10)

signupButton=Button(frame,text='Signup',font=('Times New
Roman',16,'bold'),bd=0,bg='firebrick',fg='white',
activebackground='firebrick',activeforeground='white',width=17,command=con
nect_database)
signupButton.grid(row=11,column=0,pady=10)

alreadyaccount=Label(frame,text="Don't have an account?",font=('Open
Sans','9','bold'),bg='white',fg='firebrick')
alreadyaccount.grid(row=12,column=0,sticky='w',padx=25,pady=10)

loginButton=Button(frame,text="Log in",font=('Open Sans',9,'bold underline')
,bg='white',fg='blue',bd=0,cursor='hand2',activebackground='white',activeforegr
ound='blue',command=Login)
loginButton.place(x=170,y=385)

signup_window.mainloop()
```

Employee Details Form:-

```
from tkinter import*
from tkinter import messagebox,ttk
import pymysql
import time
import os
import tempfile

class EmployeeSystem:
   def __init__(self,root):
      self.root=root
      self.root.title("Employee Payroll Management System")
      self.root.geometry("1350x700+0+0")
      self.root.config(bg="white")
      title=Label(self.root,text="Employee Payroll Management
System",font=("times new
roman",30,"bold"),bg="#262626",fg="white",padx=10)
      title.place(x=0,y=0,relwidth=1)
      btn_emp=Button(self.root,text="All
Employee's",command=self.employee_frame,font=("times new
roman",13),bg="gray",fg="white")
      btn_emp.place(x=1100,y=10,height=30,width=120)

      #===================Frame1========================
      #============Variables===============

      self.var_emp_code=StringVar()
      self.var_designation=StringVar()
      self.var_name=StringVar()
      self.var_age=StringVar()
      self.var_gender=StringVar()
      self.var_email=StringVar()
      self.var_hiredlocation=StringVar()
      self.var_dob=StringVar()
```

```
        self.var_doj=StringVar()
        self.var_proof_id=StringVar() #Adhaar Card No.
        self.var_contact=StringVar()
        self.var_status=StringVar()
        self.var_experience=StringVar()

        Frame1=Frame(self.root,bd=5,relief=RIDGE,bg="white")
        Frame1.place(x=10,y=70,width=750,height=620)
        title2=Label(Frame1,text="Employee Details",font=("times new
roman",20,),bg="lightgray",fg="black",padx=10).place(x=0,y=0,relwidth=1)

        lbl_code=Label(Frame1,text="Employee Code",font=("times new
roman",20,),bg="white",fg="black").place(x=10,y=70)
        self.txt_code=Entry(Frame1,font=("times new
roman",15),textvariable=self.var_emp_code,bg="lightyellow",fg="black")
        self.txt_code.place(x=220,y=75,width=200)

btn_search=Button(Frame1,text="Search",command=self.search,font=("times
new roman",20,),bg="gray",fg="black").place(x=440,y=72,height=30)

        #==============ROW1===============
        lbl_designation=Label(Frame1,text="Designation",font=("times new
roman",20,),bg="white",fg="black").place(x=10,y=120)
        txt_designation=Entry(Frame1,font=("times new
roman",15),textvariable=self.var_designation,bg="lightyellow",fg="black").place
(x=170,y=125,width=200)
        lbl_dob=Label(Frame1,text="D.O.B.",font=("times new
roman",20),bg="white",fg="black").place(x=390,y=120)
        txt_dob=Entry(Frame1,font=("times new
roman",15),textvariable=self.var_dob,bg="lightyellow",fg="black").place(x=520,
y=125)

        #==============ROW2===============
        lbl_name=Label(Frame1,text="Name",font=("times new
roman",20,),bg="white",fg="black").place(x=10,y=170)
```

```
        txt_name=Entry(Frame1,font=("times new
roman",15),textvariable=self.var_name,bg="lightyellow",fg="black").place(x=17
0,y=175,width=200)
        lbl_doj=Label(Frame1,text="D.O.J.",font=("times new
roman",20,),bg="white",fg="black").place(x=390,y=170)
        txt_doj=Entry(Frame1,font=("times new
roman",15),textvariable=self.var_doj,bg="lightyellow",fg="black").place(x=520,y
=175)

        #==============ROW3===============
        lbl_age=Label(Frame1,text="Age",font=("times new
roman",20,),bg="white",fg="black").place(x=10,y=220)
        txt_age=Entry(Frame1,font=("times new
roman",15),textvariable=self.var_age,bg="lightyellow",fg="black").place(x=170,y
=225,width=200)
        lbl_experience=Label(Frame1,text="Experience",font=("times new
roman",20,),bg="white",fg="black").place(x=390,y=220)
        txt_experience=Entry(Frame1,font=("times new
roman",15),textvariable=self.var_experience,bg="lightyellow",fg="black").place(
x=520,y=225)

        #==============ROW4===============
        lbl_gender=Label(Frame1,text="Gender",font=("times new
roman",20),bg="white",fg="black").place(x=10,y=270)
        txt_gender=Entry(Frame1,font=("times new
roman",15),textvariable=self.var_gender,bg="lightyellow",fg="black").place(x=1
70,y=275,width=200)
        lbl_proof=Label(Frame1,text="Proof ID",font=("times new
roman",20),bg="white",fg="black").place(x=390,y=270)
        txt_proof=Entry(Frame1,font=("times new
roman",15),textvariable=self.var_proof_id,bg="lightyellow",fg="black").place(x=
520,y=275)

        #==============ROW5===============
```

```
    lbl_email=Label(Frame1,text="Email",font=("times new
roman",20),bg="white",fg="black").place(x=10,y=320)
    txt_email=Entry(Frame1,font=("times new
roman",15),textvariable=self.var_email,bg="lightyellow",fg="black").place(x=170
,y=325,width=200)
    lbl_contact=Label(Frame1,text="Contact No.",font=("times new
roman",20,),bg="white",fg="black").place(x=390,y=320)
    txt_contact=Entry(Frame1,font=("times new
roman",15),textvariable=self.var_contact,bg="lightyellow",fg="black").place(x=5
20,y=325)

    #==============ROW6===============
    lbl_hired=Label(Frame1,text="Hired Location",font=("times new
roman",19,),bg="white",fg="black").place(x=10,y=372)
    txt_hired=Entry(Frame1,font=("times new
roman",15),textvariable=self.var_hiredlocation,bg="lightyellow",fg="black").plac
e(x=170,y=375,width=200)
    lbl_status=Label(Frame1,text="Status",font=("times new
roman",20,),bg="white",fg="black").place(x=390,y=370)
    txt_status=Entry(Frame1,font=("times new
roman",15),textvariable=self.var_status,bg="lightyellow",fg="black").place(x=52
0,y=375)

    #==============ROW7===============
    lbl_address=Label(Frame1,text="Address",font=("times new
roman",20,),bg="white",fg="black").place(x=10,y=422)
    self.txt_address=Text(Frame1,font=("times new
roman",15,),bg="lightyellow",fg="black")
    self.txt_address.place(x=170,y=425,width=550,height=150)

    #===================Frame2========================
    #============Variables===============

    self.var_month=StringVar()
    self.var_year=StringVar()
```

```
        self.var_salary=StringVar()
        self.var_totaldays=StringVar()
        self.var_absents=StringVar()
        self.var_medical=StringVar()
        self.var_providentfund=StringVar()
        self.var_convence=StringVar()
        self.var_netsalary=StringVar()

        Frame2=Frame(self.root,bd=3,relief=RIDGE,bg="white")
        Frame2.place(x=770,y=70,width=580,height=300)
        title3=Label(Frame2,text="Employee Salary Details",font=("times new
roman",20,),bg="lightgray",fg="black",padx=10).place(x=0,y=0,relwidth=1)

        lbl_month=Label(Frame2,text="Month",font=("times new
roman",18,),bg="white",fg="black").place(x=10,y=60)
        txt_month=Entry(Frame2,font=("times new
roman",15),textvariable=self.var_month,bg="lightyellow",fg="black").place(x=90
,y=62,width=100)

        lbl_year=Label(Frame2,text="Year",font=("times new
roman",18,),bg="white",fg="black").place(x=210,y=60)
        txt_year=Entry(Frame2,font=("times new
roman",15),textvariable=self.var_year,bg="lightyellow",fg="black").place(x=270,
y=62,width=100)

        lbl_salary=Label(Frame2,text="Salary",font=("times new
roman",18,),bg="white",fg="black").place(x=380,y=60)
        txt_salary=Entry(Frame2,font=("times new
roman",15),textvariable=self.var_salary,bg="lightyellow",fg="black").place(x=46
0,y=62,width=100)

        #==============ROW1===============
        lbl_days=Label(Frame2,text="Total Days",font=("times new
roman",18,),bg="white",fg="black").place(x=10,y=120)
```

```
    txt_days=Entry(Frame2,font=("times new
roman",15),textvariable=self.var_totaldays,bg="lightyellow",fg="black").place(x=
170,y=125,width=100)
    lbl_absent=Label(Frame2,text="Absents",font=("times new
roman",18,),bg="white",fg="black").place(x=300,y=120)
    txt_absent=Entry(Frame2,font=("times new
roman",15),textvariable=self.var_absents,bg="lightyellow",fg="black").place(x=4
20,y=125,width=120)

    #==============ROW2===============
    lbl_medical=Label(Frame2,text="Medical",font=("times new
roman",18,),bg="white",fg="black").place(x=10,y=150)
    txt_medical=Entry(Frame2,font=("times new
roman",15),textvariable=self.var_medical,bg="lightyellow",fg="black").place(x=1
70,y=155,width=100)
    lbl_pf=Label(Frame2,text="PF",font=("times new
roman",18,),bg="white",fg="black").place(x=300,y=150)
    txt_pf=Entry(Frame2,font=("times new
roman",15),textvariable=self.var_providentfund,bg="lightyellow",fg="black").pla
ce(x=420,y=155,width=120)

    #==============ROW3===============
    lbl_convence=Label(Frame2,text="Convence",font=("times new
roman",18,),bg="white",fg="black").place(x=10,y=180)
    txt_convence=Entry(Frame2,font=("times new
roman",15),textvariable=self.var_convence,bg="lightyellow",fg="black").place(x
=170,y=185,width=100)
    lbl_netsalary=Label(Frame2,text="Net Salary",font=("times new
roman",18,),bg="white",fg="black").place(x=300,y=180)
    txt_netsalary=Entry(Frame2,font=("times new
roman",15),textvariable=self.var_netsalary,bg="lightyellow",fg="black").place(x=
420,y=185,width=120)


btn_calculate=Button(Frame2,text="Calculate",command=self.calculate,font=("t
```

```
imes new
roman",20,),bg="orange",fg="black").place(x=150,y=225,height=30,width=120)
        self.btn_save=Button(Frame2,text="Save",command=self.add,font=("times
new roman",20,),bg="green",fg="white")
        self.btn_save.place(x=285,y=225,height=30,width=120)
        btn_clear=Button(Frame2,text="Clear",command=self.clear,font=("times
new
roman",20,),bg="gray",fg="black").place(x=420,y=225,height=30,width=120)


self.btn_update=Button(Frame2,text="Update",state=DISABLED,command=self.
update,font=("times new roman",20,),bg="blue",fg="white")
        self.btn_update.place(x=150,y=260,height=30,width=180)

self.btn_delete=Button(Frame2,text="Delete",state=DISABLED,command=self.d
elete,font=("times new roman",20,),bg="red",fg="white")
        self.btn_delete.place(x=340,y=260,height=30,width=200)

        #===================Frame3========================
        Frame3=Frame(self.root,bd=3,relief=RIDGE,bg="white")
        Frame3.place(x=770,y=380,width=580,height=310)

        #=================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="white",bd=2,relief=RIDGE)
    Cal_Frame.place(x=2,y=2,width=246,height=300)


txt_result=Entry(Cal_Frame,bg='lightyellow',textvariable=self.var_txt,font=("tim
es new roman",20,"bold"),justify=RIGHT).place(x=0,y=0,relwidth=1,height=50)

    #==========Row1==========

btn_7=Button(Cal_Frame,text='7',command=lambda:btn_click(7),font=("times
new roman",15,"bold")).place(x=0,y=52,width=60,height=60)

btn_8=Button(Cal_Frame,text='8',command=lambda:btn_click(8),font=("times
new roman",15,"bold")).place(x=61,y=52,width=60,height=60)

btn_9=Button(Cal_Frame,text='9',command=lambda:btn_click(9),font=("times
new roman",15,"bold")).place(x=122,y=52,width=60,height=60)

btn_divide=Button(Cal_Frame,text='/',command=lambda:btn_click('/'),font=("ti
mes new roman",15,"bold")).place(x=183,y=52,width=60,height=60)

    #==========Row2==========

btn_4=Button(Cal_Frame,text='4',command=lambda:btn_click(4),font=("times
new roman",15,"bold")).place(x=0,y=112,width=60,height=60)

btn_5=Button(Cal_Frame,text='5',command=lambda:btn_click(5),font=("times
new roman",15,"bold")).place(x=61,y=112,width=60,height=60)

btn_6=Button(Cal_Frame,text='6',command=lambda:btn_click(6),font=("times
new roman",15,"bold")).place(x=122,y=112,width=60,height=60)
```

```
btn_multiply=Button(Cal_Frame,text='*',command=lambda:btn_click('*'),font=(
"times new roman",15,"bold")).place(x=183,y=112,width=60,height=60)

    #==========Row3==========

btn_1=Button(Cal_Frame,text='1',command=lambda:btn_click(1),font=("times
new roman",15,"bold")).place(x=0,y=172,width=60,height=60)

btn_2=Button(Cal_Frame,text='2',command=lambda:btn_click(2),font=("times
new roman",15,"bold")).place(x=61,y=172,width=60,height=60)

btn_3=Button(Cal_Frame,text='3',command=lambda:btn_click(3),font=("times
new roman",15,"bold")).place(x=122,y=172,width=60,height=60)
    btn_subtract=Button(Cal_Frame,text='-',command=lambda:btn_click('-
'),font=("times new roman",15,"bold")).place(x=183,y=172,width=60,height=60)

    #==========Row4==========

btn_zero=Button(Cal_Frame,text='0',command=lambda:btn_click(0),font=("time
s new roman",15,"bold")).place(x=0,y=232,width=60,height=60)
    btn_clear=Button(Cal_Frame,text='C',command=clear_cal,font=("times new
roman",15,"bold")).place(x=61,y=232,width=60,height=60)

btn_add=Button(Cal_Frame,text='+',command=lambda:btn_click('+'),font=("tim
es new roman",15,"bold")).place(x=122,y=232,width=60,height=60)
    btn_equal=Button(Cal_Frame,text='=',command=result,font=("times new
roman",15,"bold")).place(x=183,y=232,width=60,height=60)

    #=================Calculator Frame============
    sal_Frame=Frame(Frame3,bg="white",bd=2,relief=RIDGE)
    sal_Frame.place(x=251,y=2,width=320,height=300)
    title_sal=Label(sal_Frame,text="Salary Receipt",font=("times new
roman",20,),bg="lightgray",fg="black",padx=10).place(x=0,y=0,relwidth=1)
```

```
        sal_Frame2=Frame(sal_Frame,bg="white",bd=2,relief=RIDGE)
        sal_Frame2.place(x=0,y=30,relwidth=1,height=230)
        self.sample=f'''\tCompany Name, XYZ\n\tAddress: Xyz, Floor4
----------------------------------------------
 Employee ID\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
 Convence\t\t:  Rs.----
 Medical\t\t:  Rs.----
 Provident Fund\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(fill=Y,side=RIGHT)

        self.txt_salary_receipt=Text(sal_Frame2,font=("times new
roman",13),bg='lightyellow',yscrollcommand=scroll_y.set)
        self.txt_salary_receipt.pack(fill=BOTH,expand=1)
        scroll_y.config(command=self.txt_salary_receipt.yview)
        self.txt_salary_receipt.insert(END,self.sample)


self.btn_print=Button(sal_Frame,text="Print",state=DISABLED,command=self.pri
nt,font=("times new roman",20,'bold'),bg="blue",fg="lightblue")
        self.btn_print.place(x=180,y=262,height=30,width=120)

        self.check_connection()
```

```
#==============================================All Functions Start
Here=================================================

    def search(self):
        try:
            con=pymysql.connect(host='localhost',user='root',password='',db='ems')
            cur=con.cursor()
            cur.execute("select * from emp_salary where
e_id=%s",(self.var_emp_code.get()))
            row=cur.fetchone()
            # print(rows)
            if row==None:
                messagebox.showerror("Error","Invalid Employee Id, please try with
another Employee ID",parent=self.root)
            else:
                # print(row)
                self.var_emp_code.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_hiredlocation.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.txt_address.delete('1.0',END)
                self.txt_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_totaldays.set(row[17])
            self.var_absents.set(row[18])
            self.var_medical.set(row[19])
            self.var_providentfund.set(row[20])
            self.var_convence.set(row[21])
            self.var_netsalary.set(row[22])

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

        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=DISABLED)
        self.btn_delete.config(state=DISABLED)
        self.btn_print.config(state=DISABLED)
        self.txt_code.config(state=NORMAL)

        self.var_emp_code.set('')
        self.var_designation.set('')
        self.var_name.set('')
        self.var_age.set('')
        self.var_gender.set('')
        self.var_email.set('')
        self.var_hiredlocation.set('')
        self.var_doj.set('')
        self.var_dob.set('')
        self.var_experience.set('')
        self.var_proof_id.set('')
```

```
        self.var_contact.set('')
        self.var_status.set('')
        self.txt_address.delete('1.0',END)
        self.var_month.set('')
        self.var_year.set('')
        self.var_salary.set('')
        self.var_totaldays.set('')
        self.var_absents.set('')
        self.var_medical.set('')
        self.var_providentfund.set('')
        self.var_convence.set('')
        self.var_netsalary.set('')
        self.txt_salary_receipt.delete('1.0',END)
        self.txt_salary_receipt.insert(END,self.sample)


    def delete(self):
        if self.var_emp_code.get()==='':
            messagebox.showerror("Error","Employee Id must be required")
        else:
            try:

con=pymysql.connect(host='localhost',user='root',password='',db='ems')
                cur=con.cursor()
                cur.execute("select * from emp_salary where
e_id=%s",(self.var_emp_code.get()))
                row=cur.fetchone()
                # print(rows)
                if row==None:
                    messagebox.showerror("Error","Invalid Employee Id, please try with
another Employee ID",parent=self.root)
                else:
                    op=messagebox.askyesno("Confirm","Do you really want to delete?")
                    # print(op)
                    if op==True:
```

```
                cur.execute("delete from emp_salary where
e_id=%s",(self.var_emp_code.get()))
                con.commit()
                con.close()
                messagebox.showinfo("Delete","Employee Record Deleted
Successfully",parent=self.root)
                self.clear()
        except Exception as ex:
            messagebox.showerror("Error",f'Error due to: {str(ex)}')


    def add(self):
        if self.var_emp_code.get()=='' or self.var_netsalary.get()=='' or
self.var_name.get()==='':
        messagebox.showerror("Error","Employee details are required")
        else:
        try:

con=pymysql.connect(host='localhost',user='root',password='',db='ems')
            cur=con.cursor()
            cur.execute("select * from emp_salary where
e_id=%s",(self.var_emp_code.get()))
            row=cur.fetchone()
            # print(rows)
            if row!=None:
                messagebox.showerror("Error","This Employee Id has already
available in our record,try again with another Id",parent=self.root)
            else:
                cur.execute("insert into emp_salary values
(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s
)",
                (
                    self.var_emp_code.get(),
                    self.var_designation.get(),
                    self.var_name.get(),
```

```
                self.var_age.get(),
                self.var_gender.get(),
                self.var_email.get(),
                self.var_hiredlocation.get(),
                self.var_doj.get(),
                self.var_dob.get(),
                self.var_experience.get(),
                self.var_proof_id.get(),
                self.var_contact.get(),
                self.var_status.get(),
                self.txt_address.get('1.0',END),
                self.var_month.get(),
                self.var_year.get(),
                self.var_salary.get(),
                self.var_totaldays.get(),
                self.var_absents.get(),
                self.var_medical.get(),
                self.var_providentfund.get(),
                self.var_convence.get(),
                self.var_netsalary.get(),
                self.var_emp_code.get()+".txt"
            )
            )
            con.commit()
            con.close()
            file_=open('Salary_Receipt/'+str(self.var_emp_code.get())+".txt",'w')
            file_.write(self.txt_salary_receipt.get('1.0',END))
            file_.close()
            messagebox.showinfo("Success","Record Added Successfully")
            self.btn_print.config(state=NORMAL)
        except Exception as ex:
            messagebox.showerror("Error",f'Error due to: {str(ex)}')


    def update(self):
```

```
        if self.var_emp_code.get()=='' or self.var_netsalary.get()=='' or
self.var_name.get()==='':
            messagebox.showerror("Error","Employee details are required")
        else:
            try:

con=pymysql.connect(host='localhost',user='root',password='',db='ems')
                cur=con.cursor()
                cur.execute("select * from emp_salary where
e_id=%s",(self.var_emp_code.get()))
                row=cur.fetchone()
                # print(rows)
                if row==None:
                    messagebox.showerror("Error","This Employee Id is invalid, try again
with valid Employee Id",parent=self.root)
                else:
                    cur.execute("UPDATE `emp_salary` SET
`designation`=%s,`name`=%s,`age`=%s,`gender`=%s,`email`=%s,`hr_location`=%s
,`doj`=%s,`dob`=%s,`experience`=%s,`proof_id`=%s,`contact`=%s,`status`=%s,`ad
dress`=%s,`month`=%s,`year`=%s,`basic_salary`=%s,`total_days`=%s,`absent_day
s`=%s,`medical`=%s,`provident_fund`=%s,`convence`=%s,`net_salary`=%s,`salary
_receipt`=%s WHERE `e_id`=%s",
                    (
                        self.var_designation.get(),
                        self.var_name.get(),
                        self.var_age.get(),
                        self.var_gender.get(),
                        self.var_email.get(),
                        self.var_hiredlocation.get(),
                        self.var_doj.get(),
                        self.var_dob.get(),
                        self.var_experience.get(),
                        self.var_proof_id.get(),
                        self.var_contact.get(),
                        self.var_status.get(),
```

```
                self.txt_address.get('1.0',END),
                self.var_month.get(),
                self.var_year.get(),
                self.var_salary.get(),
                self.var_totaldays.get(),
                self.var_absents.get(),
                self.var_medical.get(),
                self.var_providentfund.get(),
                self.var_convence.get(),
                self.var_netsalary.get(),
                self.var_emp_code.get()+".txt",
                self.var_emp_code.get()
            )
            )
            con.commit()
            con.close()
            file_=open('Salary_Receipt/'+str(self.var_emp_code.get())+".txt",'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 calculate(self):
        if self.var_month.get()== '' or self.var_year.get()== '' or
self.var_salary.get()== '' or self.var_totaldays.get()== '' or
self.var_absents.get()== '' or self.var_medical.get()== '' or
self.var_providentfund.get()== '' or self.var_convence.get()== '':
            messagebox.showerror('Error','All fields are required')
        else:
            # self.var_netsalary.set("Result")
            # 35000/31==1752
            # 31-10=21*1752
            per_day=int(self.var_salary.get())/int(self.var_totaldays.get())
```

```
        work_day=int(self.var_totaldays.get())-int(self.var_absents.get())
        sal_=per_day*work_day
        deduct=int(self.var_medical.get())+int(self.var_providentfund.get())
        addition=int(self.var_convence.get())
        net_sal=sal_-deduct+addition
        self.var_netsalary.set(str(round(net_sal,2)))
        #==================Update the receipt==============
        new_sample=f'''\tCompany Name, XYZ\n\tAddress: Xyz, Floor4
----------------------------------------------
 Employee ID\t\t:  {self.var_emp_code.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_totaldays.get()}
 Total Present\t\t:  {str(int(self.var_totaldays.get())-int(self.var_absents.get()))}
 Total Absent\t\t:  {self.var_absents.get()}
 Convence\t\t:  Rs.{self.var_convence.get()}
 Medical\t\t:  Rs.{self.var_medical.get()}
 Provident Fund\t\t:  Rs.{self.var_providentfund.get()}
 Gross Payment\t\t:  Rs.{self.var_salary.get()}
 Net Salary\t\t:  Rs.{self.var_netsalary.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)
        self.btn_print.config(state=NORMAL)

    def check_connection(self):
        try:
        con=pymysql.connect(host='localhost',user='root',password='',db='ems')
        cur=con.cursor()
        cur.execute("select * from emp_salary")
```

```
            rows=cur.fetchall()
            # print(rows)
        except Exception as ex:
            messagebox.showerror("Error",f'Error due to: {str(ex)}')

    def show(self):
        try:
            con=pymysql.connect(host='localhost',user='root',password='',db='ems')
            cur=con.cursor()
            cur.execute("select * from emp_salary")
            rows=cur.fetchall()
            # print(rows)
            self.employee_tree.delete(*self.employee_tree.get_children())
            for row in rows:
                self.employee_tree.insert('',END,values=row)
            con.close()
        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 Management System")
        self.root2.geometry("1000x500+120+100")
        self.root2.config(bg="white")
        title=Label(self.root2,text="All Employee Details",font=("times new
roman",30,"bold"),bg="#262626",fg="white",padx=10).pack(side=TOP,fill=X)
        self.root2.focus_force()

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

        self.employee_tree=ttk.Treeview(self.root2,columns=('e_id', 'designation',
'name', 'age', 'gender', 'email', 'hr_location', 'doj', 'dob', 'experience', 'proof_id',
```

```
'contact', 'status', 'address', 'month', 'year', 'basic_salary', 'total_days',
'absent_days', 'medical', 'provident_fund', 'convence', 'net_salary',
'salary_receipt'),yscrollcommand=scrolly.set,xscrollcommand=scrollx.set)
        self.employee_tree.heading('e_id',text='E_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='HiredLocation')
        self.employee_tree.heading('doj',text='D.O.J.')
        self.employee_tree.heading('dob',text='D.O.B.')
        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('total_days',text='Total Days')
        self.employee_tree.heading('absent_days',text='Absent Days')
        self.employee_tree.heading('medical',text='Medical')
        self.employee_tree.heading('provident_fund',text='Provident Fund')
        self.employee_tree.heading('convence',text='Convence')
        self.employee_tree.heading('net_salary',text='Net Salary')
        self.employee_tree.heading('salary_receipt',text='SalaryReceipt')
        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=100)
        self.employee_tree.column('age',width=100)
        self.employee_tree.column('gender',width=100)
```

```
        self.employee_tree.column('email',width=100)
        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=500)
        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('total_days',width=100)
        self.employee_tree.column('absent_days',width=100)
        self.employee_tree.column('medical',width=100)
        self.employee_tree.column('provident_fund',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()

    def print(self):
        file_=tempfile.mktemp(".txt")
        open(file_,'w').write(self.txt_salary_receipt.get('1.0',END))
        os.startfile(file_,'print')

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

Testing and Implementation

System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences. Testing is the process of executing the program with the intent of finding errors and missing operation.

UNIT TESTING:-

Unit testing focuses first on modules, independently of one another, to locate errors. This enables, to detect errors in coding and logic that are contained within each module. The various controls are tested to ensure that each performs its action as required.

Integration Testing

Integration testing is a systematic testing to discover errors associated within the interface. The objective is to take unit tested modules and build a program structure.

USER ACCEPTANCE TESTING

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the system users at time of developing and making changes whenever required.

Future Scope

It is easy to extend the system that we have proposed. A person could see any of the issued, unissued or all the employees according to his/her will. In future we can implement some features for “Employee Payroll Management System” project. In the system its possible to categorize salary according to attendance. Also help to provide Provident fund to the employees.

CONCLUSION

Employee Payroll Management System is a Customize and user-friendly software for Employees. It has been designed to automate, manage and look after the overall processing employees salary. It is capable of managing employee details, employee salary details, attendance details ,etc. Employee Payroll Management System is a Customize and user-friendly software for Employees which provide employees information, salary information, present and absent information.

Employee Payroll Management System is offering a maximum of stability, cost-effectiveness and usability. It provides the most flexible and adaptable standards management system software solutions for employees.

BIBLOGRAPHY

- PROJECT WEBSITES:-
  - www.pythonproject.com
  - www.codewithharry.com
  - www.webcode.com

- REFERENCE:-
  - Python Programming For Beginners:- “ANTHONY ADAMS”
  - Python Programming:- “JOHN ZELLE”