PYTHON COURSE MINIPROJECT

**Payroll Management System**

HARIPRASATH L



Project Report submitted in Partial fulfilment requirement for the Proficient

Certificate Course

Done By

**HARIPRASATH L**

Under the Guidance of Approved By

**VIDHYA SAKTHI VEL**



**ABOUT THE PROJECT:**

The **Payroll Management System** is an automated software solution designed to streamline the process of managing employee payroll within an organization. It simplifies complex tasks such as calculating employee wages, handling tax deductions, processing bonuses, and generating payslips. The system is intended to reduce manual errors, improve accuracy, and save time by automating payroll processes. It also ensures compliance with tax laws and regulations by automatically calculating deductions and generating tax reports. Additionally, the system provides secure storage of employee data, ensures timely salary disbursements, and supports efficient management of employee benefits and deductions. This system is essential for organizations of any size, as it not only enhances operational efficiency but also improves employee satisfaction by ensuring timely and accurate payment processing. Ultimately, the Payroll Management System offers a comprehensive solution to manage payroll tasks with ease and reliability

**Project Prerequisites**:

To build this project, we will need the following libraries:

1. Tkinter– To make the GUI

1. mysql connector

**TKINTER**

* Graphical User Interfaces (GUIs) are essential components of modern software applications, providing users with intuitive ways to interact with complex systems.
* Tkinter, a Python library, offers a convenient tool kit for developing GUIs with ease and flexibility.
* In this project, we focus on harnessing the capabilities of Tkinter to build an interactive GUI application tailored to specific needs.
* The project involves the design and implementation of a user-friendly interface using Tkinter's extensive widget set, which includes buttons, labels, entry fields, and more.
* By leveraging Tkinter's event-driven programming model, we aim to create are responsive and dynamic interface that enhances user engagement.
* The GUI application will be developed with a focus on usability and aesthetics, ensuring that users can easily navigate through various functionalities and perform tasks efficiently.
* Through thoughtful layout design and intuitive user interactions, we strive to delivery interface that minimizes cognitive load and maximizes user satisfaction.
* By developing an interactive GUI application using Tkinter, this project aims to showcase the versatility and effectiveness of Tkinter in creating user-friendly software interfaces.
* The resulting application will demonstrate the power of Python and Tkinter in building GUIs that combine functionality, usability, and aesthetics to meet the needs of diverse user groups.

**1.LABEL**

A Label widget in Tkinter is used to display text or image graphical user interface (GUI).Labels are static elements, meaning they do not accept user input. They are typically used for displaying information, titles, or any other static content on the GUI.

SYNTAX : label = Label (parent, options)

**2. BUTTON**

A Button widget in Tkinter is used to create clickable button graphical user interface (GUI). Buttons are interactive elements that perform a specific action when clicked by the user.

SYNTAX : button = Button (parent, options)

**3.FRAME**

A Frame is a container widget used to group and organize other widgets within a window. It helps in structuring complex sections inside the main window. The frame can also be nested within other frames for more complex layouts.

SYNTAX : frame = Frame (parent, options)

**4.ENTRY**

An Entry widget in Tkinter is used to create a single-line text entry field where users can input text or data. Entry widgets allow users to type in alphanumeric characters or numbers and are commonly used for accepting user input in forms, search boxes, or data entry fields.

SYNTAX : entry = Entry (parent, options)

**5.MESSAGEBOX**

Messagebox in Tkinter is a dialog box that displays a message to the user and typically includes buttons for user interaction, such as confirming an action or providing additional information.

Tkinter provides a messagebox module that contains functions for creating

and customizing message boxes.

The message box module provides several functions for creating different types of messageboxes, such as showinfo(), showwarning(),and showerror()

SYNTAX : result = messagebox.function\_name

(title, message, options)

**6.Tree Widget**

**A Tree Widget** is a container that displays hierarchical data in a tree-like structure, where items can have parent-child relationships. It is useful for displaying complex structures like file systems, organizational charts, or any other data that has a tree hierarchy. The tree widget supports operations such as expanding/collapsing nodes, selecting items, and adding/removing items dynamically.

tree = Tree(parent, options)

**SOURCE CODE:**

import tkinter as tk

from tkinter import ttk

from tkinter import Label, Entry, Button, Spinbox

from tkinter import messagebox, Toplevel

from tkinter.font import Font

from PIL import Image, ImageTk

import mysql.connector

from mysql.connector import Error

def connect\_db():

return mysql.connector.connect(

host="localhost",

user="root",

password="",

database="payroll\_management"

)

root = tk.Tk()

root.title("Payroll System")

root.geometry("900x700")

root.resizable(False, False)

image\_path = r"C:\Users\BeastHari\Desktop\Python\ProjectLesson\payroll\_bg.jpg"

image = Image.open(image\_path)

bg\_image = ImageTk.PhotoImage(image)

font\_1 = Font(family="times", size=14, weight="bold")

font\_2 = Font(family="times", size=25, weight="bold")

font\_3 = Font(family="times", size=18)

def main():

main\_frame =ttk.Frame(root)

main\_frame.pack(fill="both", expand=True)

bg\_label = Label(main\_frame, image=bg\_image)

bg\_label.place(relheight=1, relwidth=1)

def emp\_entry():

main\_frame.destroy()

Employee\_Entry()

def emp\_details():

main\_frame.destroy()

Emp\_details()

def emp\_update\_call():

main\_frame.destroy()

emp\_update()

def fetch\_employee\_details(emp\_number):

try:

conn = connect\_db()

cursor = conn.cursor()

cursor.execute("SELECT \* FROM employee\_file WHERE number = %s", (emp\_number,))

employee = cursor.fetchone()

conn.close()

return employee

except Error as e:

print(f"Database error: {e}")

return None

def search\_employee():

Emp = Toplevel(root)

Emp.geometry("400x350")

Emp.config(bg="lightblue")

Emp.title("Employee Details")

emp\_number = search\_emp.get()

employee = fetch\_employee\_details(emp\_number)

if employee:

labnew = Label(Emp, text="Employee Details", font=("times", 15, "bold"), fg="white", bg="#2c3e80", padx=10,pady=10)

labnew.pack()

details = (f"Number: {employee[0]}\nName: {employee[1]}\nSex: {employee[2]}\nDOB: {employee[3]}\nDOJ: {employee[4]}\nDesignation: {employee[5]}"

f"\nBasic Salary: {employee[6]}\nPhone: {employee[7]}\nAddress: {employee[8]}")

lab\_details = Label(Emp, text=details, font=("times", 12,"bold"), fg="black", bg="lightblue", padx=10, pady=10)

lab\_details.pack()

else:

messagebox.showerror("Error", "Employee not found!")

btnnew = Button(Emp, text="Exit", font=("times", 15, "bold"), width=10, padx=0, pady=0, bg="red", fg="white",command=Emp.destroy)

btnnew.pack(padx=10, pady=10)

def delete():

dele = Toplevel(root)

dele.geometry("200x150")

dele.config(bg="black")

dele.title("Employee Delete")

def emp\_delete():

emp\_number = emp\_delete\_entry.get()

if not emp\_number:

messagebox.showwarning("Input Error", "Please enter an Employee Number.")

return

try:

conn = connect\_db()

if conn is None:

raise Exception("Database connection failed.")

cursor = conn.cursor()

cursor.execute("DELETE FROM monthly\_pay\_file WHERE number = %s", (emp\_number,))

cursor.execute("DELETE FROM employee\_file WHERE number = %s", (emp\_number,))

conn.commit()

if cursor.rowcount > 0:

messagebox.showinfo("Success", f"Employee {emp\_number} deleted successfully.")

else:

messagebox.showwarning("Not Found", f"No employee found with number {emp\_number}.")

conn.close()

dele.destroy()

except Error as e:

messagebox.showerror("Database Error", str(e))

except Exception as e:

messagebox.showerror("Error", str(e))

ttk.Label(dele, text="Delete Employee", font=font\_3).pack()

ttk.Label(dele, text="Employee No:").pack()

emp\_delete\_entry = Entry(dele)

emp\_delete\_entry.pack()

delete\_button = ttk.Button(dele, text="Delete", command=emp\_delete)

delete\_button.pack(pady=5)

exit\_button = ttk.Button(dele, text="Exit", command=dele.destroy)

exit\_button.pack(pady=5)

ttk.Label(main\_frame, text="Payroll Management", background="blue", padding=10, font=font\_2).pack()

ttk.Label(main\_frame, text="Employee.No:", font=font\_1).pack()

search\_emp = ttk.Entry(main\_frame)

search\_emp.pack()

submit\_search = Button(main\_frame, text="Search", font=font\_1, command=search\_employee).pack()

emp\_entry\_button = Button(main\_frame, text="Entry", font=font\_2, command=emp\_entry, padx=10, pady=10)

emp\_entry\_button.place(x=230, y=200)

emp\_update\_button = Button(main\_frame, text="Modify", font=font\_2, command=emp\_update\_call, padx=10, pady=10)

emp\_update\_button.place(x=380, y=200)

emp\_details\_button = Button(main\_frame, text="Details", font=font\_2, command=emp\_details, padx=10, pady=10)

emp\_details\_button.place(x=550, y=200)

emp\_delete\_button = Button(main\_frame, text="Delete", font=font\_2, command=delete, padx=10, pady=10)

emp\_delete\_button.place(x=385, y=300)

emp\_details\_button = Button(main\_frame, text="Exit", font=font\_2, command=quit, padx=10, pady=10)

emp\_details\_button.place(x=395, y=500)

root.mainloop()

def Employee\_Entry():

def submit\_form():

salary\_dict = {

"Driver": 30000,

"Manager": 60000,

"Security": 25000

}

number = number\_entry.get()

name = name\_entry.get()

sex = sex\_var.get()

dob = f"{DOB\_d.get()}/{DOB\_m.get()}/{DOB\_y.get()}"

doj = f"{DOJ\_d.get()}/{DOJ\_m.get()}/{DOJ\_y.get()}"

designation = designation\_entry.get()

basic\_salary = salary\_dict.get(designation, 0)

phone = phone\_entry.get()

address = address\_entry.get("1.0", tk.END)

conn = connect\_db()

cursor = conn.cursor()

cursor.execute(

"INSERT INTO employee\_file (number, name, sex, Dob, Doj, designation, basic\_salary, phone, address) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)",

(number, name, sex, dob, doj, designation, basic\_salary, phone, address))

conn.commit()

conn.close()

messagebox.showinfo("Success", "Employee added successfully!")

def clear\_form():

number\_entry.delete(0, tk.END)

name\_entry.delete(0, tk.END)

sex\_var.set(None)

DOB\_d.delete(0, tk.END)

DOB\_m.delete(0, tk.END)

DOB\_y.delete(0, tk.END)

DOJ\_d.delete(0, tk.END)

DOJ\_m.delete(0, tk.END)

DOJ\_y.delete(0, tk.END)

designation\_entry.delete(0, tk.END)

phone\_entry.delete(0, tk.END)

address\_entry.delete(0, tk.END)

def back\_form():

form\_frame.destroy()

main()

form\_frame = ttk.Frame(root, padding="10", style='TFrame')

form\_frame.pack(fill="both", expand=True)

style = ttk.Style()

style.configure('TFrame', background='blue')

ttk.Label(form\_frame, text="Employee Entry", font=font\_2, padding=10, background="black",

foreground="white").pack()

ttk.Label(form\_frame, text="Employee Number:", font=font\_1, background="blue").pack()

number\_entry = ttk.Entry(form\_frame)

number\_entry.pack()

ttk.Label(form\_frame, text="Name:", font=font\_1, background="blue").pack()

name\_entry = ttk.Entry(form\_frame)

name\_entry.pack()

ttk.Label(form\_frame, text="Sex:", font=font\_1, background="blue").pack()

sex\_var = tk.StringVar()

ttk.Radiobutton(form\_frame, text="Male", variable=sex\_var, value="Male").pack()

ttk.Radiobutton(form\_frame, text="Female", variable=sex\_var, value="Female").pack()

ttk.Label(form\_frame, text="Date of Birth(dd/mm/yyyy):", font=font\_1, background="blue").pack()

DOB\_d = Spinbox(form\_frame, from\_=1, to=31, width=2, font=font\_1)

DOB\_d.place(x=380, y=250)

DOB\_m = Spinbox(form\_frame, from\_=1, to=12, width=2, font=font\_1)

DOB\_m.pack()

DOB\_y = Spinbox(form\_frame, from\_=1980, to=2020, width=5, font=font\_1)

DOB\_y.place(x=462, y=250)

ttk.Label(form\_frame, text="Date of Joining(dd/mm/yyyy):", font=font\_1, background="blue").pack()

DOJ\_d = Spinbox(form\_frame, from\_=1, to=31, width=2, font=font\_1)

DOJ\_d.place(x=380, y=302)

DOJ\_m = Spinbox(form\_frame, from\_=1, to=12, width=2, font=font\_1)

DOJ\_m.pack()

DOJ\_y = Spinbox(form\_frame, from\_=1980, to=2020, width=5, font=font\_1)

DOJ\_y.place(x=462, y=302)

ttk.Label(form\_frame, text="Designation:", font=font\_1, background="blue").pack()

designation\_entry = ttk.Combobox(form\_frame, width=20, state="readonly")

designation\_entry["values"] = ("Driver", "Manager", "Security")

designation\_entry.current(1)

designation\_entry.pack()

ttk.Label(form\_frame, text="Phone:", font=font\_1, background="blue").pack()

phone\_entry = ttk.Entry(form\_frame)

phone\_entry.pack()

ttk.Label(form\_frame, text="Address:", font=font\_1, background="blue").pack()

address\_entry = tk.Text(form\_frame, height=5, width=20)

address\_entry.pack()

submit\_button = Button(form\_frame, text="Submit",command=submit\_form,font=font\_1)

submit\_button.pack()

clear\_button = Button(form\_frame, text="Clear", command=clear\_form,font=font\_1)

clear\_button.pack()

back\_button = Button(form\_frame, text="Back", command=back\_form, font=font\_1)

back\_button.pack()

root.mainloop()

def emp\_update():

update\_frame = ttk.Frame(root,style="TFrame")

update\_frame.pack(fill="both", expand=1)

style = ttk.Style()

style.configure('TFrame', background='orange')

ttk.Label(update\_frame, text="Employee Update", font=font\_2, padding=10, background="black",

foreground="white").pack()

ttk.Label(update\_frame, text="Employee Number:", font=font\_1, background="orange").pack()

number\_entry = ttk.Entry(update\_frame)

number\_entry.pack()

ttk.Label(update\_frame, text="Name:", font=font\_1, background="orange").pack()

name\_entry = ttk.Entry(update\_frame)

name\_entry.pack()

ttk.Label(update\_frame, text="Sex:", font=font\_1, background="orange").pack()

sex\_var = tk.StringVar()

ttk.Radiobutton(update\_frame, text="Male", variable=sex\_var, value="Male").pack()

ttk.Radiobutton(update\_frame, text="Female", variable=sex\_var, value="Female").pack()

ttk.Label(update\_frame, text="Date of Birth(dd/mm/yyyy):", font=font\_1, background="orange").pack()

DOB\_d = Spinbox(update\_frame, from\_=1, to=31, width=2, font=font\_1)

DOB\_d.place(x=390, y=250)

DOB\_m = Spinbox(update\_frame, from\_=1, to=12, width=2, font=font\_1)

DOB\_m.pack()

DOB\_y = Spinbox(update\_frame, from\_=1980, to=2020, width=5, font=font\_1)

DOB\_y.place(x=472, y=250)

ttk.Label(update\_frame, text="Date of Joining(dd/mm/yyyy):", font=font\_1, background="orange").pack()

DOJ\_d = Spinbox(update\_frame, from\_=1, to=31, width=2, font=font\_1)

DOJ\_d.place(x=390, y=302)

DOJ\_m = Spinbox(update\_frame, from\_=1, to=12, width=2, font=font\_1)

DOJ\_m.pack()

DOJ\_y = Spinbox(update\_frame, from\_=1980, to=2020, width=5, font=font\_1)

DOJ\_y.place(x=472, y=302)

ttk.Label(update\_frame, text="Designation:", font=font\_1, background="orange").pack()

designation\_entry = ttk.Combobox(update\_frame, width=20, state="readonly")

designation\_entry["values"] = ("Driver", "Manager", "Security")

designation\_entry.current(1)

designation\_entry.pack()

ttk.Label(update\_frame, text="Phone:", font=font\_1, background="orange").pack()

phone\_entry = ttk.Entry(update\_frame)

phone\_entry.pack()

ttk.Label(update\_frame, text="Address:", font=font\_1, background="orange").pack()

address\_entry = tk.Text(update\_frame, height=5, width=20)

address\_entry.pack()

def clear\_form():

number\_entry.delete(0, tk.END)

name\_entry.delete(0, tk.END)

sex\_var.set(None)

DOB\_d.delete(0, tk.END)

DOB\_m.delete(0, tk.END)

DOB\_y.delete(0, tk.END)

DOJ\_d.delete(0, tk.END)

DOJ\_m.delete(0, tk.END)

DOJ\_y.delete(0, tk.END)

designation\_entry.delete(0, tk.END)

phone\_entry.delete(0, tk.END)

address\_entry.delete(0, tk.END)

def update\_employee():

salary\_dict = {

"Driver": 30000,

"Manager": 60000,

"Security": 25000

}

emp\_number = number\_entry.get()

emp\_name = name\_entry.get()

emp\_sex = sex\_var.get()

emp\_dob = f"{DOB\_d.get()}/{DOB\_m.get()}/{DOB\_y.get()}"

emp\_doj = f"{DOJ\_d.get()}/{DOJ\_m.get()}/{DOJ\_y.get()}"

emp\_designation = designation\_entry.get()

emp\_basic\_salary = salary\_dict.get(emp\_designation, 0)

emp\_phone = phone\_entry.get()

emp\_address = address\_entry.get("1.0", tk.END)

if not emp\_number or not emp\_name:

messagebox.showwarning("Input Error", "Please enter both Employee Number and Name.")

return

try:

conn = connect\_db()

cursor = conn.cursor()

# Update the employee details in the database

cursor.execute(

"UPDATE employee\_file SET sex = %s, Dob=%s, Doj=%s,designation=%s, basic\_salary=%s, phone=%s, address=%s WHERE number = %s",

(emp\_sex, emp\_dob, emp\_doj, emp\_designation, emp\_basic\_salary, emp\_phone, emp\_address, emp\_number))

cursor.execute(

"UPDATE employee\_file SET sex = %s, Dob=%s, Doj=%s,designation=%s, basic\_salary=%s, phone=%s, address=%s WHERE name = %s",

(emp\_sex, emp\_dob, emp\_doj, emp\_designation, emp\_basic\_salary, emp\_phone, emp\_address, emp\_name))

conn.commit()

conn.close()

messagebox.showinfo("Success", "Employee details updated successfully.")

except Error as e:

messagebox.showerror("Database Error", str(e))

def back():

update\_frame.destroy()

main()

update\_button = Button(update\_frame, text="Update", command=update\_employee, font=font\_1)

update\_button.pack(pady=10)

clear\_button = Button(update\_frame, text="Clear", command=clear\_form, font=font\_1)

clear\_button.pack()

back\_button = Button(update\_frame, text="Back", command=back, font=font\_1)

back\_button.pack(pady=10)

def Emp\_details():

style = ttk.Style()

style.configure("TNotebook", background="purple")

notebook = ttk.Notebook(root)

notebook.pack(fill="both",expand=True)

style = ttk.Style()

style.configure('TFrame', background='green')

frame\_emp = ttk.Frame(notebook, height=500, width=700,padding=20,style="TFrame")

frame\_emp.pack()

frame\_salary\_details = ttk.Frame(notebook, height=500, width=700,padding=20,style="TFrame")

frame\_salary\_details.pack()

frame\_salary = ttk.Frame(notebook, height=500, width=700,padding=20,style="TFrame")

frame\_salary.pack()

notebook.add(frame\_emp, text="Employees Details")

notebook.add(frame\_salary\_details,text="Monthly Pay")

notebook.add(frame\_salary, text="Salary Slip")

def back():

notebook.destroy()

main()

def display\_employee\_details():

tree = ttk.Treeview(frame\_emp, columns=("ID", "Name","sex","dob","doj","Designation", "Salary"), show='headings')

tree.heading("ID", text="Employee.No")

tree.heading("Name", text="Name")

tree.heading("sex", text="Sex")

tree.heading("dob", text="DOB")

tree.heading("doj", text="DOJ")

tree.heading("Designation", text="Designation")

tree.heading("Salary", text="Basic Salary")

tree.pack()

tree.column("ID", width=100)

tree.column("Name", width=150)

tree.column("sex", width=50)

tree.column("dob", width=100)

tree.column("doj", width=100)

tree.column("Designation", width=150)

tree.column("Salary", width=100)

def show\_all\_employees():

for item in tree.get\_children():

tree.delete(item)

try:

conn = connect\_db()

cursor = conn.cursor()

cursor.execute("SELECT \* FROM employee\_file")

employees = cursor.fetchall()

for employee in employees:

tree.insert("", "end", values=employee)

conn.close()

except Error as e:

messagebox.showerror("Database Error", str(e))

btn\_show\_all = ttk.Button(frame\_emp, text="Back", command=back)

btn\_show\_all.pack(pady=10)

show\_all\_employees()

display\_employee\_details()

def salary\_details():

tree = ttk.Treeview(frame\_salary\_details, columns=("ID", "Name", "Designation", "Salary","Gross","Deduction","Net"),

show='headings')

tree.heading("ID", text="Employee.No")

tree.heading("Name", text="Name")

tree.heading("Designation", text="Designation")

tree.heading("Salary", text="Basic Salary")

tree.heading("Gross", text="Gross")

tree.heading("Deduction", text="Deduction")

tree.heading("Net", text="Net")

tree.pack()

tree.column("ID", width=100)

tree.column("Name", width=120)

tree.column("Designation", width=100)

tree.column("Gross", width=100)

tree.column("Deduction", width=100)

tree.column("Net", width=100)

def show\_all\_employees():

for item in tree.get\_children():

tree.delete(item)

try:

conn = connect\_db()

cursor = conn.cursor()

query = """

SELECT e.number, e.name, designation, mp.basic\_salary\_e , mp.gross, mp.deduction, mp.net

FROM employee\_file e

JOIN monthly\_pay\_file mp ON e.number = mp.number

"""

cursor.execute(query)

employees = cursor.fetchall()

for employee in employees:

tree.insert("", "end", values=employee)

conn.close()

except Error as e:

messagebox.showerror("Database Error", str(e))

show\_all\_employees()

salary\_details()

def calculate\_salary(employee\_number, days\_worked):

try:

conn = connect\_db()

cursor = conn.cursor()

cursor.execute("SELECT name, basic\_salary FROM employee\_file WHERE number = %s", (employee\_number,))

result = cursor.fetchone()

if result:

name, basic\_salary = result

max\_days\_in\_month = 30

actual\_basic = (basic\_salary \* days\_worked) / max\_days\_in\_month

da = actual\_basic \* 0.55

hra = actual\_basic \* 0.35

conveyance = actual\_basic \* 0.15

gross = actual\_basic + da + hra + conveyance

deductions = (basic\_salary\*5)/100

net = gross - deductions

cursor.execute(

"INSERT INTO monthly\_pay\_file (number, name, no\_of\_days, basic\_salary\_e, deduction, DA, HRA, conveyence, gross, net) VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",

(employee\_number, name, days\_worked, actual\_basic, deductions, da, hra, conveyance, gross, net))

conn.commit()

messagebox.showinfo("Success", "Salary calculated and stored successfully.")

else:

messagebox.showwarning("Not Found", "Employee not found.")

except Error as e:

messagebox.showerror("Database Error", str(e))

def salary\_input():

newl11 = Label(frame\_salary\_details,text="Calculate the Salary",font=font\_3,bg="green")

newl11.pack()

newl1 = Label(frame\_salary\_details, text="Employee Number:", font=font\_1,bg="green")

newl1.pack()

employee\_num = ttk.Entry(frame\_salary\_details)

employee\_num.pack()

newl = Label(frame\_salary\_details, text="No.of days worked:", font=font\_1,bg="green")

newl.pack()

days\_work = ttk.Entry(frame\_salary\_details)

days\_work.pack()

def go():

try:

employee\_number = employee\_num.get()

days\_worked = int(days\_work.get())

calculate\_salary(employee\_number, days\_worked)

except ValueError:

messagebox.showwarning("Input Error", "Please enter a valid number of days worked.")

go\_salary = Button(frame\_salary\_details, text="Go", font=font\_1, command=go)

go\_salary.pack()

btn\_back = ttk.Button(frame\_salary\_details, text="Back", command=back)

btn\_back.pack(pady=10)

salary\_input()

def fetch\_employee\_details(emp\_number):

try:

conn = connect\_db()

cursor = conn.cursor()

cursor.execute("SELECT \* FROM monthly\_pay\_file WHERE number = %s", (emp\_number,))

employee = cursor.fetchone()

conn.close()

return employee

except Error as e:

print(f"Database error: {e}")

return None

def salary\_slip():

emp\_number = search\_emp.get()

employee = fetch\_employee\_details(emp\_number)

if employee:

labnew = Label(frame\_salary, text="Employee Salary Slip", font=("times", 15, "bold"), fg="white", bg="#2c3e80", padx=10,

pady=10)

labnew.pack()

details = (

f"Number: {employee[0]} Name: {employee[1]}\nBasic: {employee[3]} Deduction: {employee[6]}\nDA: {employee[5]}\nHRA: {employee[6]}\nConveyance: {employee[7]}"

f"\nGross: {employee[8]} Net: {employee[9]}")

lab\_details = Label(frame\_salary, text=details, font=("times", 12, "bold"), fg="black", bg="lightblue", padx=10,

pady=10)

lab\_details.pack()

else:

messagebox.showerror("Error", "Employee not found!")

ttk.Label(frame\_salary, text="Employee.No:", font=font\_1,background="green").pack()

search\_emp = ttk.Entry(frame\_salary)

search\_emp.pack()

submit\_search = Button(frame\_salary, text="Search", font=font\_1, command=salary\_slip)

submit\_search.pack()

btn\_back = ttk.Button(frame\_salary, text="Back", command=back)

btn\_back.pack(pady=10)

main()

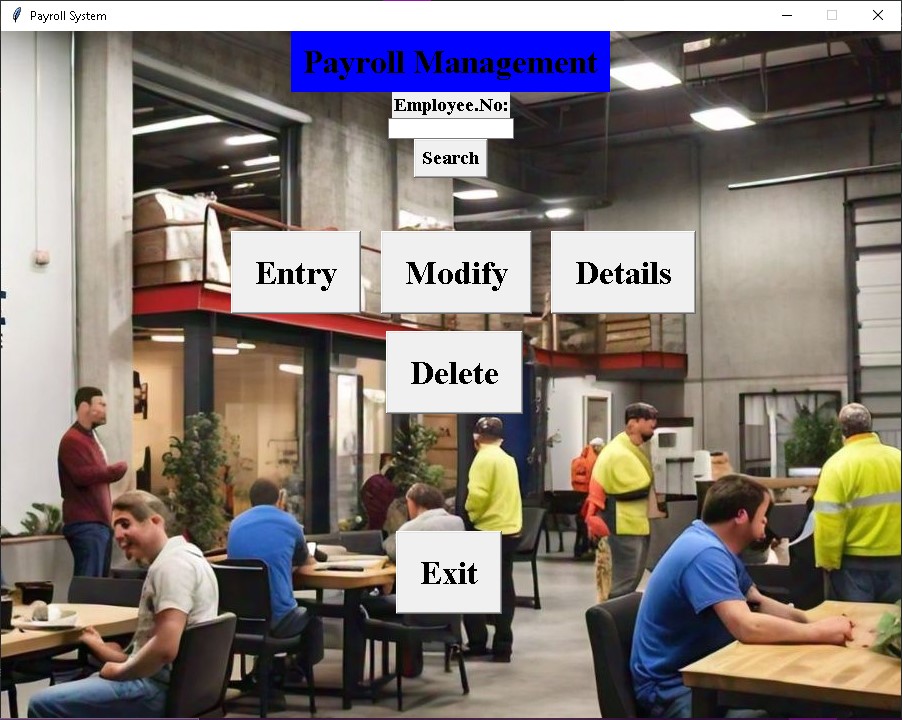
**Project Overview:** The Payroll Management System is a software application designed to automate the process of managing employee payroll within an organization. It simplifies the calculation and distribution of wages, deductions, bonuses, and taxes, ensuring accuracy and efficiency. The system aims to streamline payroll processes, reduce human error, and improve compliance with tax regulations.

**Key Features:**

1. **Employee Management**: Store and manage employee details, including personal information, salary, and position.
2. **Salary Calculation**: Automatically calculate wages based on hourly rates, salaries, overtime, and bonuses.
3. **Tax Calculation**: Automatically apply tax deductions based on local tax laws.
4. **Payslip Generation**: Generate and distribute digital payslips to employees.
5. **Deductions and Benefits**: Track deductions such as insurance, retirement contributions, and other employee benefits.
6. **Reporting**: Generate reports on payroll expenses, tax reports, and compliance records

**Project Objective:** The objective of the Payroll Management System is to reduce the time and resources spent on manual payroll tasks, increase accuracy, ensure timely payments, and maintain compliance with relevant financial regulations. It is ideal for organizations of all sizes looking to improve their payroll operations and enhance employee satisfaction.

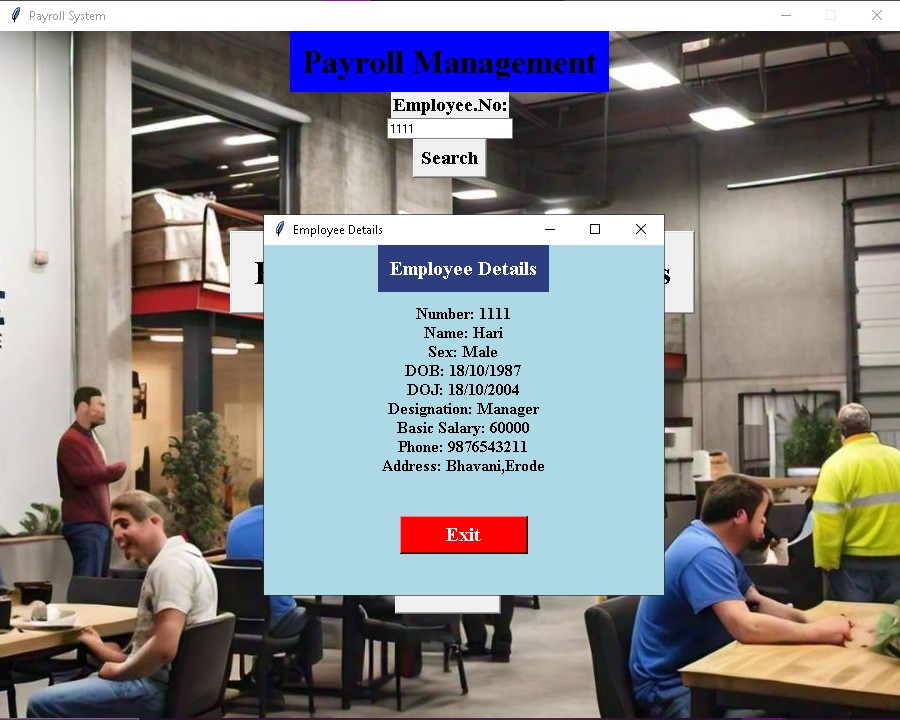
**MAIN WINDOW**

****

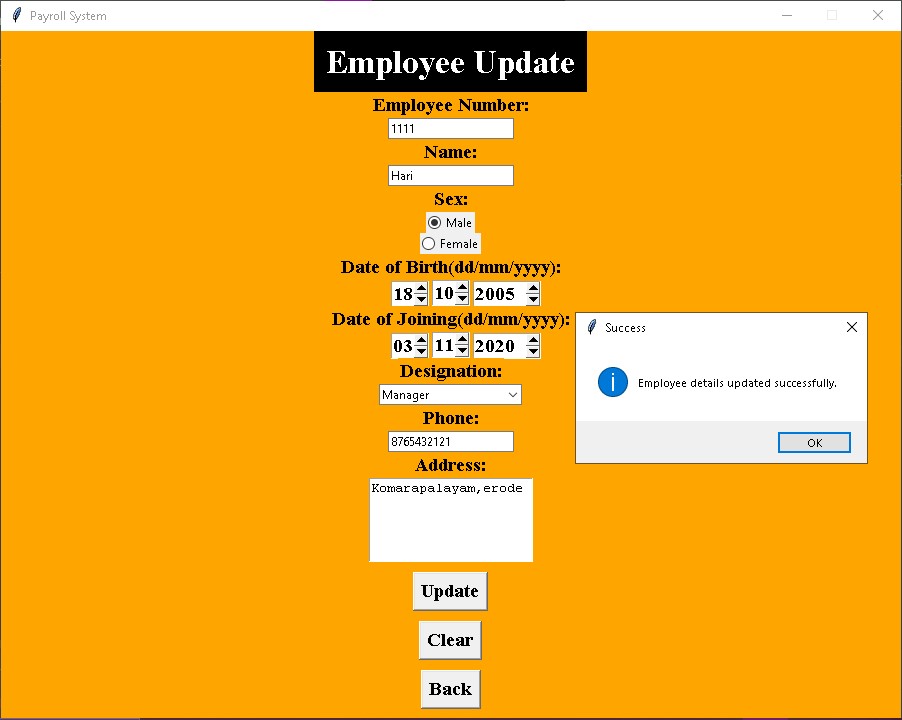
**ENTRY**



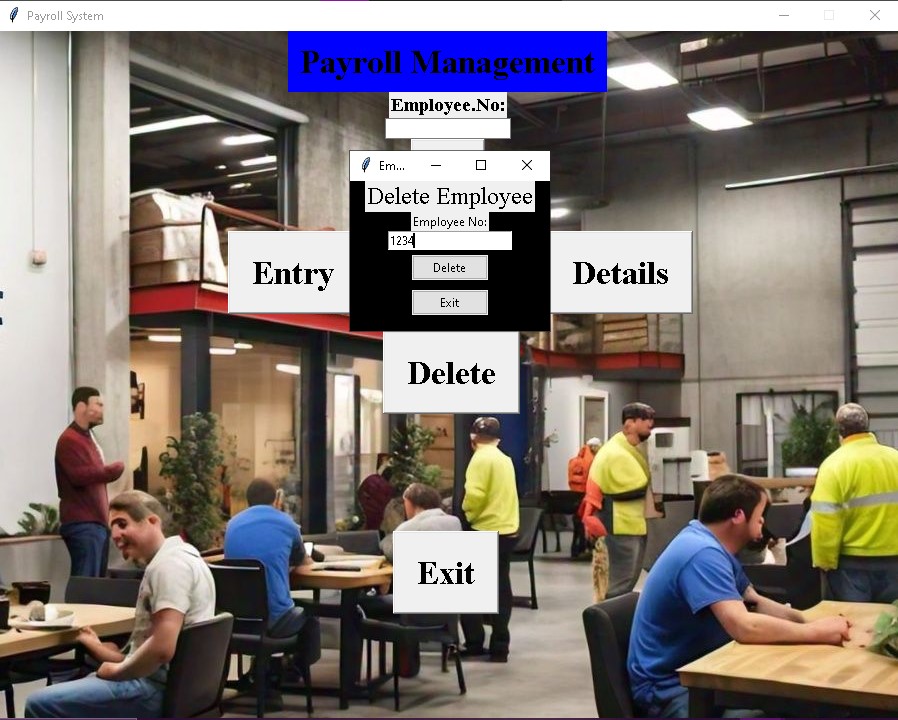
**SEARCH**



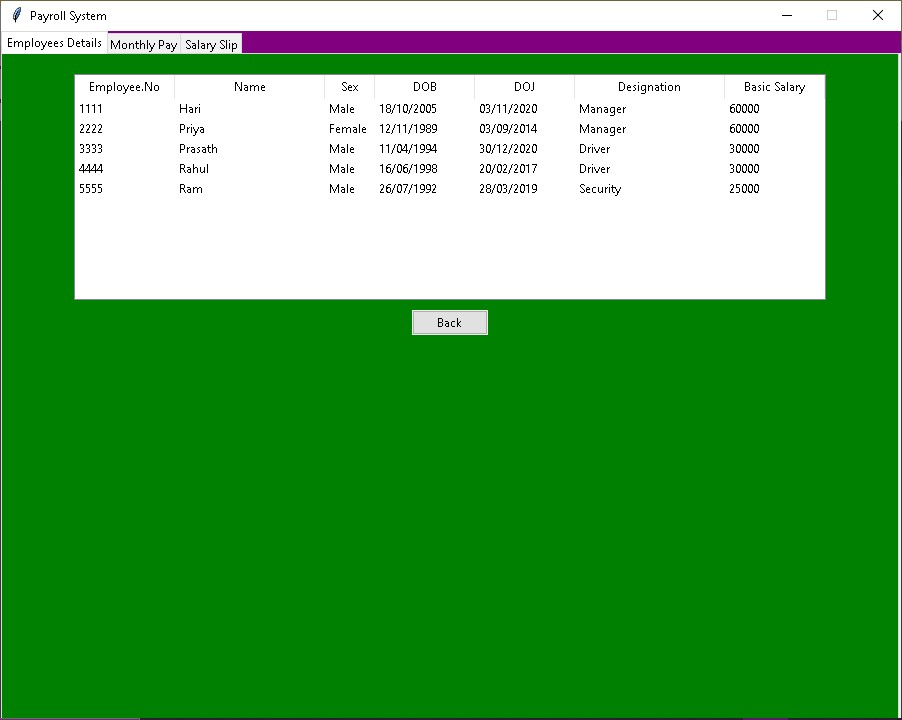
**UPDATE**



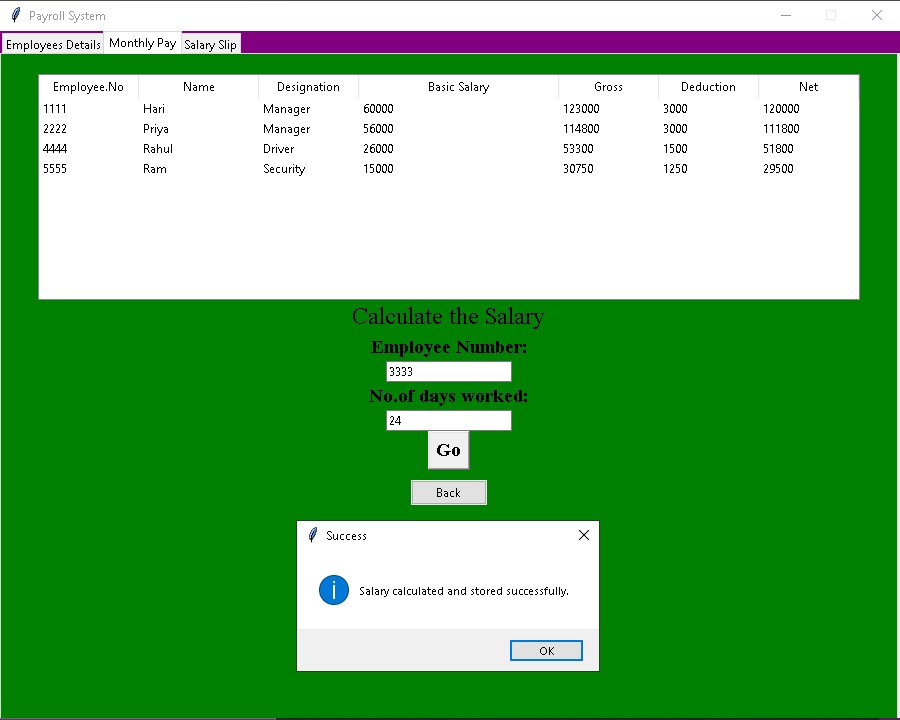
**DELETE**



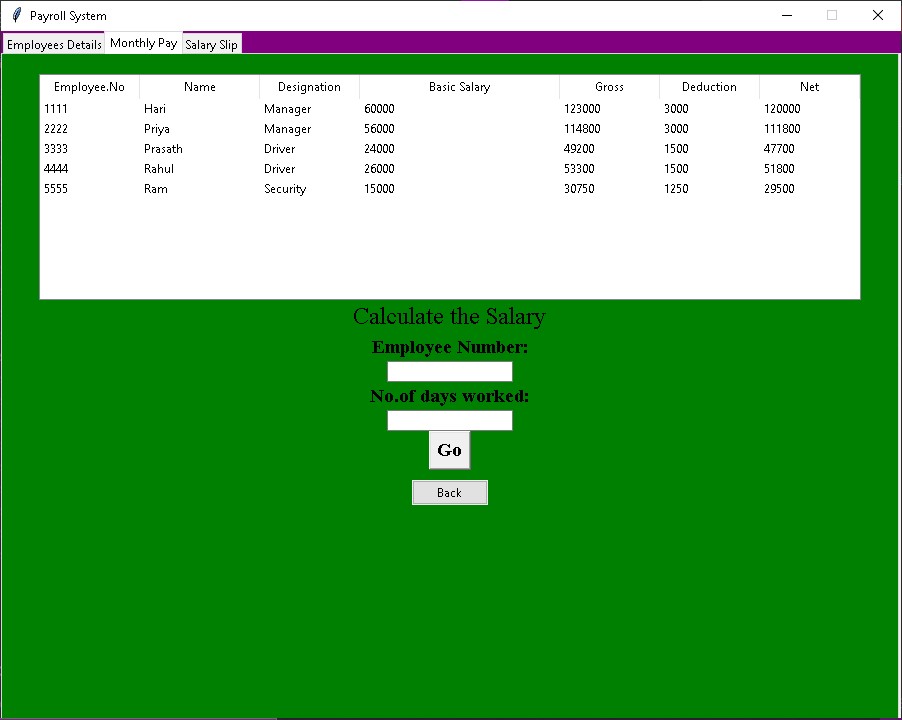
**EMPLOYEE LIST**



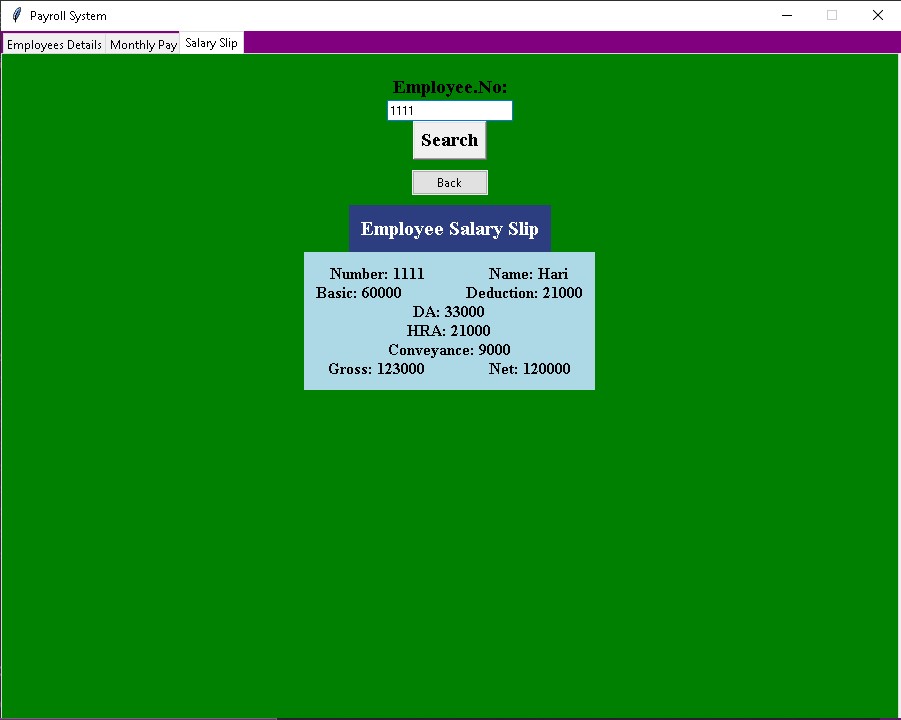
**SALARY CALCULATION**



**EMPLOYEE SALARY LIST**

****

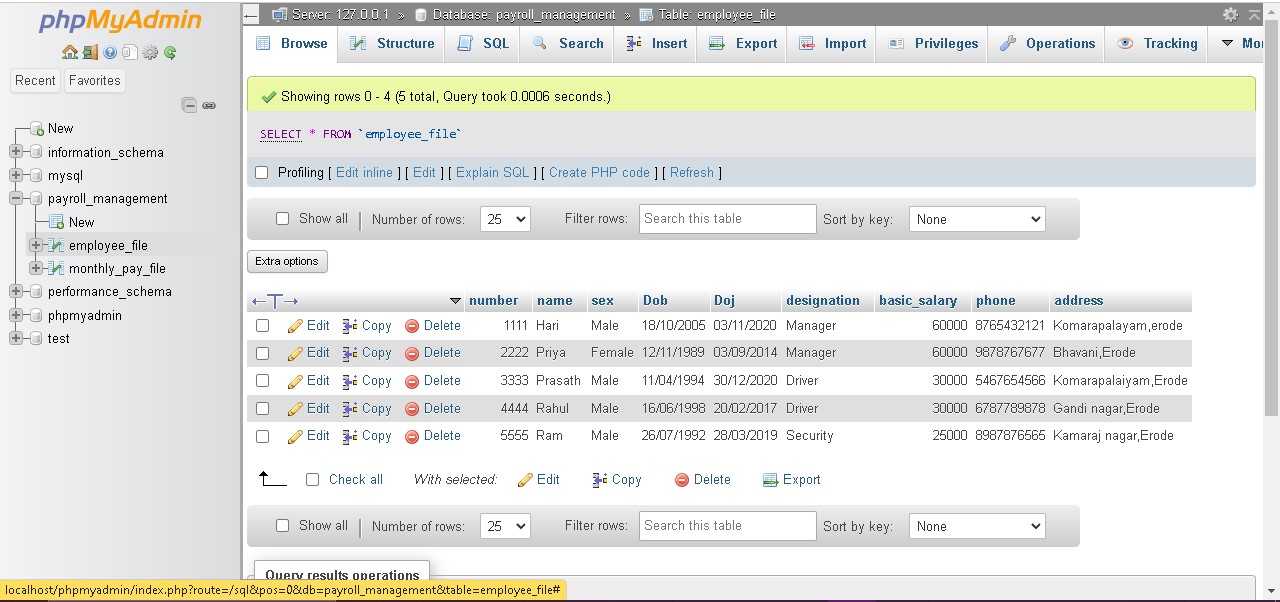
**SALARY SLIP 1**

****

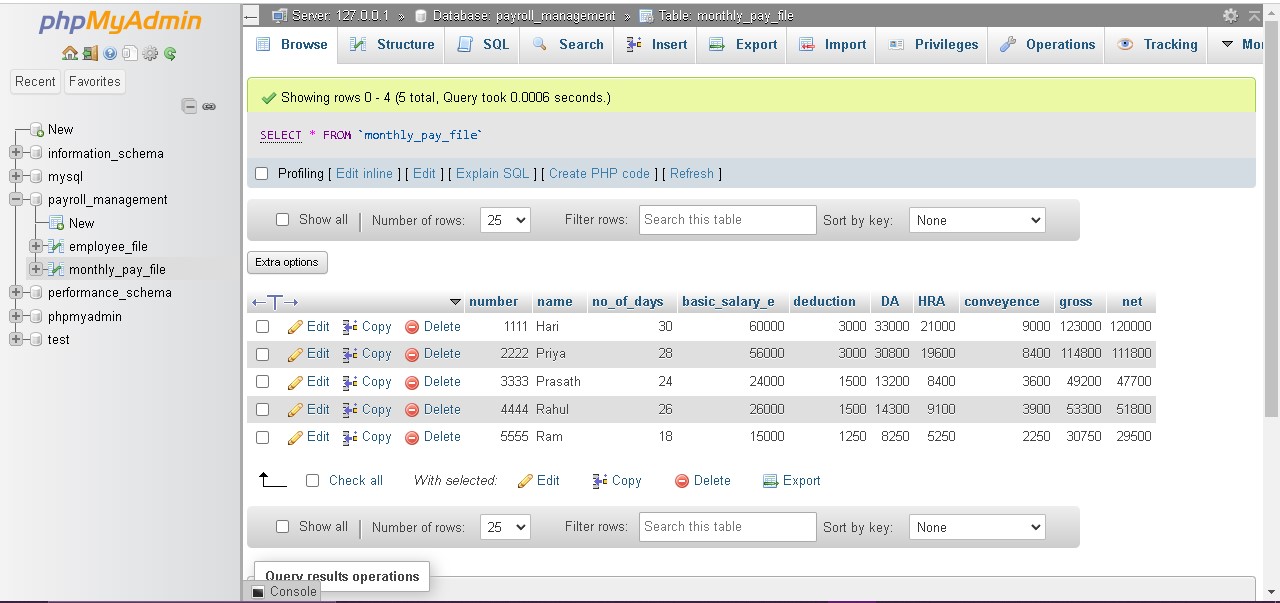
**SALARY SLIP 2**

****

**DATABASE EMPLOYEES**

****

**DATABASE MONTHLY PAY**

****

**Conclusion:**

the **Payroll Management System** is a highly effective and efficient solution for automating the payroll process within organizations. By eliminating manual calculations and paperwork, the system reduces errors, saves time, and ensures that employees are paid accurately and on time. The system's ability to handle complex tasks such as tax calculations, deductions, and benefits management, along with its user-friendly interface, makes it a valuable tool for HR and finance departments. Moreover, the system's robust reporting capabilities and compliance with legal regulations ensure that organizations meet their legal obligations with minimal effort. Overall, the Payroll Management System significantly enhances operational efficiency, improves data security, and fosters better employee satisfaction, making it an indispensable tool for modern organizations.