Report On

Employee Management System

Submitted in partial fulfillment of the requirements of the Course project in
Semester IV of Second Year Computer Engineering

by
Nishant Tanaji Bhandigare (Roll No. 05)
Vipul Naresh Bhoir (Roll No. 06)
Mrudul Parag Chaudhari (Roll No. 11)
Abhinav Bhimrao Desai (Roll No. 13)

Supervisor
Prof. Smita Jawale

University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

(2022-23)

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

CERTIFICATE

This is to certify that the project entitled “Employee Management System” is a bonafide work of "Nishant Bhandigare (Roll No. 05), Vipul Bhoir (Roll No. 06), Mrudul Chaudhari (Roll No. 11), Abhinav Desai (Roll No. 13)" submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester IV of Second Year Computer Engineering.

Supervisor

Prof. Smita Jawale

Prof. Name Surname		
Internal Examiner		External Examiner
Dr Megha Trivedi Head of Department		Dr. H.V. Vankudre Principal

Abstract :

An Employee Management System is a software application designed to help organizations manage their employee data, track employee attendance, and handle other human resource tasks efficiently. This project aims to develop an Employee Management System using the Python programming language and the Tkinter GUI toolkit.

The application will provide an easy-to-use interface for the HR department to manage employee information such as personal details, job information, and performance reviews.

The system will be developed using object-oriented programming principles and will store data in a backend database for easy retrieval and analysis. The application will also incorporate data visualization tools to provide HR managers with easy-to-understand reports on employee performance and attendance.

Overall, the Employee Management System using tkinter aims to improve the efficiency of human resource management, reduce errors in employee data management, and provide managers with valuable insights into their employees' performance.

Contents : **Page No.**

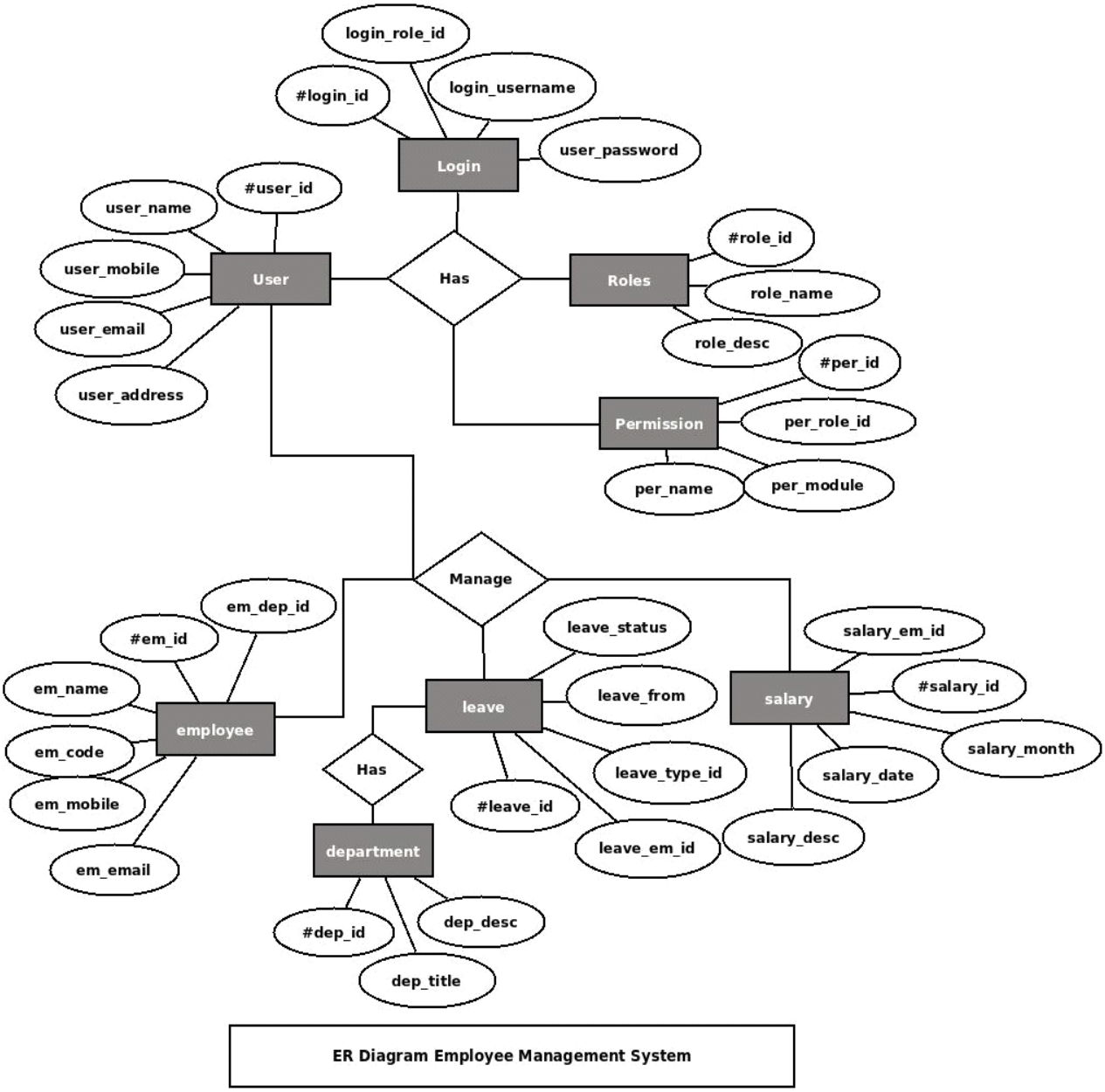
Problem Statement :

The current manual employee management system used by many organizations is prone to errors, time-consuming and lacks the necessary features required to manage the ever-increasing workforce. Manually managing employee data, attendance, performance reviews, and other human resource tasks can lead to inefficiencies, inaccuracies, and increased costs. Additionally, managers lack real-time access to accurate employee data, making it difficult to make informed decisions.

To address these challenges, an Employee Management System is required to automate the management of employee data, attendance, and performance reviews. The system should provide a user-friendly interface for HR managers to manage employee data efficiently. The system should also provide real-time access to accurate employee data, enabling managers to make informed decisions.

Therefore, the problem statement for the project on Employee Management System is to design and develop a software application that will automate the management of employee data. The application should be easy to use, efficient, accurate, and provide real-time access to employee data. The system should be able to store and retrieve employee data, generate reports, and provide managers with valuable insights into employee performance.

ER diagram:


login_role_id
#login_id
login_username
Login
user_password
#user_id
user_name
user_mobile
User
user_email
user_address
Has
Roles
#role_id
role_name
role_desc
#per_id
Permission
per_role_id
per_name
per_module
Manage
em_dep_id
#em_id
em_name
employee
em_code
em_mobile
em_email
leave_status
leave
leave_from
leave_type_id
#leave_id
leave_em_id
Has
department
#dep_id
dep_desc
dep_title
salary_em_id
salary
#salary_id
salary_month
salary_date
salary_desc
ER Diagram Employee Management System

Module Description :

Tkinter:

Tkinter is a standard Python library used for developing graphical user interfaces (GUIs). It provides a range of pre-built widgets, such as buttons, labels, text boxes, and menus, that can be used to create user interfaces for desktop applications. Tkinter also provides an event-driven programming model, where user actions such as button clicks or key presses can be handled by the application. This makes it easy to create interactive GUIs that respond to user input.
In the context of retail management systems, Tkinter can be used to create a user-friendly interface for managing tasks such as inventory management, sales tracking, employee management, and customer relationship management. Tkinter's pre-built widgets can be customized to fit the specific needs of the retail store, and its event-driven programming model can be used to handle user input and update the application in real-time.
Overall, Tkinter is a versatile and powerful library for developing desktop applications with graphical user interfaces, making it a great choice for building retail management systems.

db:

db is the created module used to perform operations on query processing using SQL and database language. Actually db is the file we have created to perform many operations like inset ,delete, update and clear.in this we have created a class database and have inserted it into the main file that we can access that functions in main file.
We can update and insert new entry in the system as well as we can delete the employee as per our requirements.

Hardware and Software Requirements:

Functional Requirements:

- Operating system : Windows/Ubuntu
- VS code
- Python 3.10
- MySQL

Important Modules :

- sql-connector
- Tkinter

Code:

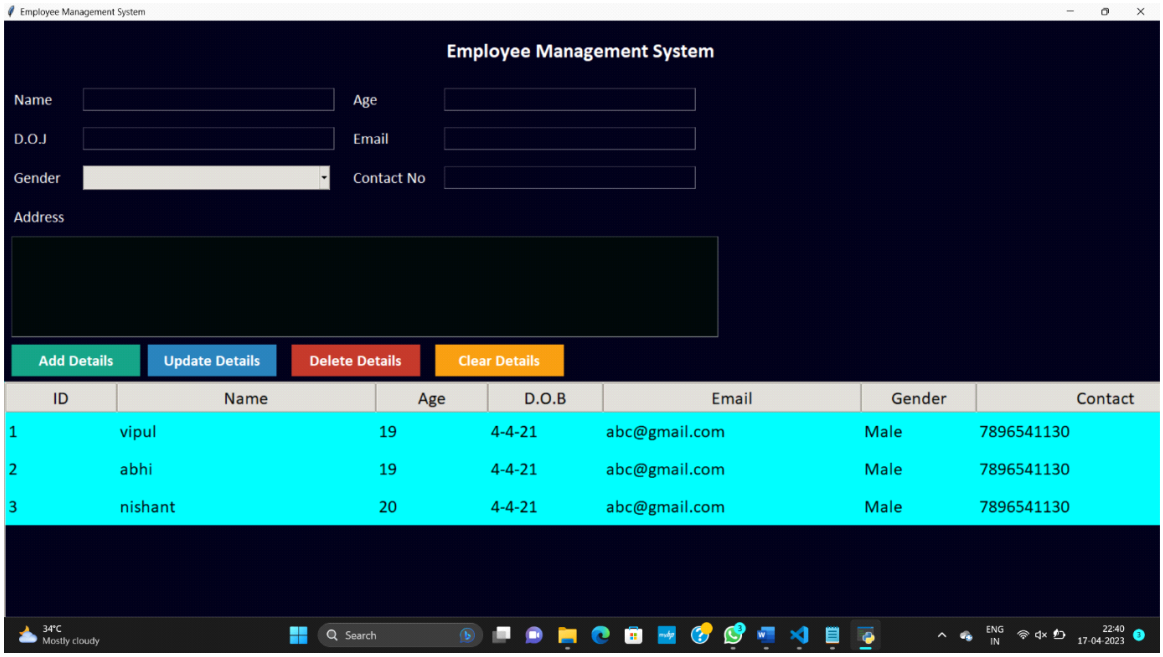
```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from db import Database

db = Database("Employee.db")
root = Tk()
root.title("Employee Management System")
root.geometry("1920x1080+0+0")
root.config(bg="aqua")
root.state("zoomed")

name = StringVar()
age = StringVar()
doj = StringVar()
gender = StringVar()
email = StringVar()
contact = StringVar()

# Entries Frame
entries_frame = Frame(root, bg="#02011c")
entries_frame.pack(side=TOP, fill=X)
title = Label(entries_frame, text="Employee Management System", font=("Calibri", 20, "bold"), bg="#02011c",
fg="white")
title.grid(row=0, column=3, padx=10, pady=20, sticky="w")

lblName = Label(entries_frame, text="Name", font=("Calibri", 16), bg="#02011c", fg="white")
lblName.grid(row=1, column=0, padx=10, pady=10, sticky="w")
txtName = Entry(entries_frame, textvariable=name, font=("Calibri", 16), width=30,bg="#02011c", fg="white")
txtName.grid(row=1, column=1, padx=10, pady=10, sticky="w")

lblAge = Label(entries_frame, text="Age", font=("Calibri", 16), bg="#02011c", fg="white")
lblAge.grid(row=1, column=2, padx=10, pady=10, sticky="w")
txtAge = Entry(entries_frame, textvariable=age, font=("Calibri", 16), width=30,bg="#02011c", fg="white")
txtAge.grid(row=1, column=3, padx=10, pady=10, sticky="w")

lbldoj = Label(entries_frame, text="D.O.J", font=("Calibri", 16), bg="#02011c", fg="white")
lbldoj.grid(row=2, column=0, padx=10, pady=10, sticky="w")
txtDoj = Entry(entries_frame, textvariable=doj, font=("Calibri", 16), width=30,bg="#02011c", fg="white")
txtDoj.grid(row=2, column=1, padx=10, pady=10, sticky="w")

lblEmail = Label(entries_frame, text="Email", font=("Calibri", 16), bg="#02011c", fg="white")
lblEmail.grid(row=2, column=2, padx=10, pady=10, sticky="w")
txtEmail = Entry(entries_frame, textvariable=email, font=("Calibri", 16), width=30,bg="#02011c", fg="white")
txtEmail.grid(row=2, column=3, padx=10, pady=10, sticky="w")

lblGender = Label(entries_frame, text="Gender", font=("Calibri", 16), bg="#02011c", fg="white")
lblGender.grid(row=3, column=0, padx=10, pady=10, sticky="w")
comboGender = ttk.Combobox(entries_frame, font=("Calibri", 16), width=28, textvariable=gender,
state="readonly")
comboGender['values'] = ("Male", "Female")
comboGender.grid(row=3, column=1, padx=10, sticky="w")

lblContact = Label(entries_frame, text="Contact No", font=("Calibri", 16), bg="#02011c", fg="white")
lblContact.grid(row=3, column=2, padx=10, pady=10, sticky="w")
```

```
txtContact = Entry(entries_frame, textvariable=contact, font=("Calibri", 16), width=30,bg="#02011c",
fg="white")
txtContact.grid(row=3, column=3, padx=10, sticky="w")

lblAddress = Label(entries_frame, text="Address", font=("Calibri", 16), bg="#02011c", fg="white")
lblAddress.grid(row=4, column=0, padx=10, pady=10, sticky="w")

txtAddress = Text(entries_frame, width=85, height=5, font=("Calibri", 16),bg="#00080a",fg="white")
txtAddress.grid(row=5, column=0, columnspan=4, padx=10, sticky="w")


def getData(event):
    selected_row = tv.focus()
    data = tv.item(selected_row)
    global row
    row = data["values"]
    #print(row)
    name.set(row[1])
    age.set(row[2])
    doj.set(row[3])
    email.set(row[4])
    gender.set(row[5])
    contact.set(row[6])
    txtAddress.delete(1.0, END)
    txtAddress.insert(END, row[7])

def dispalyAll():
    tv.delete(*tv.get_children())
    for row in db.fetch():
        tv.insert("", END, values=row)


def add_employee():
    if txtName.get() == "" or txtAge.get() == "" or txtDoj.get() == "" or txtEmail.get() == "" or
comboGender.get() == "" or txtContact.get() == "" or txtAddress.get(
            1.0, END) == "":
        messagebox.showerror("Erorr in Input", "Please Fill All the Details")
        return
    db.insert(txtName.get(),txtAge.get(), txtDoj.get() , txtEmail.get() ,comboGender.get(), txtContact.get(),
txtAddress.get(
            1.0, END))
    messagebox.showinfo("Success", "Record Inserted")
    clearAll()
    dispalyAll()

def update_employee():
    if txtName.get() == "" or txtAge.get() == "" or txtDoj.get() == "" or txtEmail.get() == "" or
comboGender.get() == "" or txtContact.get() == "" or txtAddress.get(
            1.0, END) == "":
        messagebox.showerror("Erorr in Input", "Please Fill All the Details")
        return
    db.update(row[0],txtName.get(), txtAge.get(), txtDoj.get(), txtEmail.get(), comboGender.get(),
txtContact.get(),
              txtAddress.get(
                  1.0, END))
    messagebox.showinfo("Success", "Record Update")
    clearAll()
    dispalyAll()

def delete_employee():
```

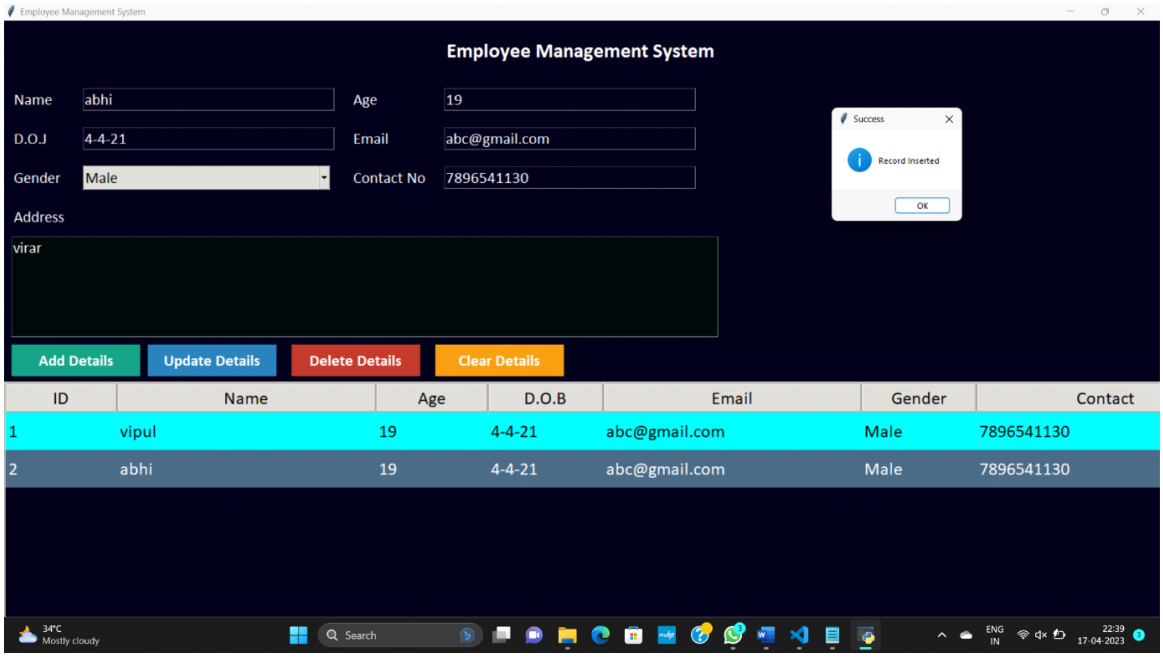
```
 db.remove(row[0])
    clearAll()
    dispalyAll()
def clearAll():
    name.set("")
    age.set("")
    doj.set("")
    gender.set("")
    email.set("")
    contact.set("")
    txtAddress.delete(1.0, END)
btn_frame = Frame(entries_frame, bg="#02011c")
btn_frame.grid(row=6, column=0, columnspan=4, padx=10, pady=10, sticky="w")
btnAdd = Button(btn_frame, command=add_employee, text="Add Details", width=15, font=("Calibri", 16,
"bold"), fg="white",
              bg="#16a085",activebackground="white", activeforeground="#16a085", bd=0).grid(row=0,
column=0)
btnEdit = Button(btn_frame, command=update_employee, text="Update Details", width=15, font=("Calibri",
16, "bold"),
                 fg="white", bg="#2980b9",activebackground="white", activeforeground="#2980b9",
                 bd=0).grid(row=0, column=1, padx=10)
btnDelete = Button(btn_frame, command=delete_employee, text="Delete Details", width=15, font=("Calibri",
16, "bold"),
                   fg="white", bg="#c0392b",activebackground="white", activeforeground="#c0392b",
                   bd=0).grid(row=0, column=2, padx=10)
btnClear = Button(btn_frame, command=clearAll, text="Clear Details", width=15, font=("Calibri", 16, "bold"),
fg="white",
                 bg="#f39c12",activebackground="white", activeforeground="#f39c12",
                 bd=0).grid(row=0, column=3, padx=10)

# Table Frame
tree_frame = Frame(root, bg="#02011c")
tree_frame.place(x=0, y=480, width=1980, height=520)
style = ttk.Style()
style.theme_use('clam')
style.configure("mystyle.Treeview", font=('Calibri', 18),
                rowheight=50,fieldbackground= "#02011c", background= "aqua")  # Modify the font of the body
style.configure("mystyle.Treeview.Heading", font=('Calibri', 18))  # Modify the font of the headings
tv = ttk.Treeview(tree_frame, columns=(1, 2, 3, 4, 5, 6, 7, 8), style="mystyle.Treeview")
tv.heading("1", text="ID")
tv.column("1", width=5)
tv.heading("2", text="Name")
tv.heading("3", text="Age")
tv.column("3", width=5)
tv.heading("4", text="D.O.B")
tv.column("4", width=10)
tv.heading("5", text="Email")
tv.heading("6", text="Gender")
tv.column("6", width=10)
tv.heading("7", text="Contact")
tv.heading("8", text="Address")
tv['show'] = 'headings'
tv.bind("<ButtonRelease-1>", getData)
tv.pack(fill=X)

dispalyAll()
root.mainloop()
```

Output and User Interface:

Insert:-



Update & Delete:-



Results And Conclusion :

The system has proven to be effective in reducing the time and effort required to manage employee data. The user-friendly interface provides HR managers with an easy-to-use platform for managing employee data and generating reports. The system has also reduced errors in employee data management, making it easier for managers to make informed decisions.

In conclusion, the Employee Management System using tkinter is a powerful software application that streamlines HR processes, reduces errors, and provides valuable insights into employee data. The system enables HR managers to make informed decisions, improve employee productivity and drive business growth. Overall, the system has proven to be an effective tool for managing employees, and it is recommended for any organization looking to streamline its HR processes.

References :

- https://docs.python.org/3/library/tkinter.html
- https://www.tutorialspoint.com/python/python_gui_programming.htm#:~:text=Tkinter%20is%20the%20standard%20GUI,Tkinter%20is%20an%20easy%20task
- https://www.geeksforgeeks.org/python-gui-tkinter/
- https://pythonprogramming.net/python-3-tkinter-basics-tutorial/