EMPLOYEE MANAGEMENT SYSTEM

MINI PROJECT REPORT

Submitted by

PRASANTH D 220701199

RAHGUL S 220701210

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2023 - 24

BONAFIDE CERTIFICATE

Certified that this project report "**EMPLOYEE MANAGEMENT SYSTEM**"

is the bonafide work of **"RAHGUL (220701210), PRASANTH D**

(220701199)"

who carried out the project work under my supervision.

Submitted for the Practical Examination held on

SIGNATURE

Mrs.K.maheshmeena

Assistant Professor (SG) ,
Computer Science and Engineering ,
Rajalakshmi Engineering College(Autonomous) ,
Thandalam, Chennai - 602 105

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

An employee management system is a software application that streamlines the tasks involved in managing a workforce.

It allows to efficiently add, delete, and update employee information. The system can also include features to clear input fields, track time and attendance, and manage payroll. Employee management systems can help to improve productivity and reduce administrative costs

The intuitive interface facilitates easy interaction, allowing users to manage employee information seamlessly.

TABLE OF CONTENTS

1. Introduction

1.1 Introduction

An Employee Management System (EMS) is a crucial tool for any organization aiming to efficiently manage its workforce. Leveraging the power of Python and Tkinter, we have developed a robust and user-friendly EMS that streamlines various HR functions such as employee data management, attendance tracking, and performance monitoring. By integrating Python's versatility with Tkinter's graphical capabilities, our EMS ensures a seamless and efficient experience, enhancing productivity and fostering better organizational management.

1.2 Objectives

The objectives of the Employee Management System include:

Attendance Tracking: EMS tracks employee attendance accurately, providing a reliable record for HR management.

Stronger Compliance: Ensure accurate record-keeping for legal and regulatory requirements

Scalability and Customization: Design the system to be scalable and customizable, allowing it to grow with the organization and adapt to specific business requirements.

1.3 Modules

The Employee Management System comprises several key modules:

User Interface Module:

This module, named main, is responsible for creating the graphical user interface (GUI) using the Tkinter library.It defines the layout and appearance of the GUI elements, including labels, entry fields, buttons, and a table view.Event handling functions for user interactions, such as adding, updating, and deleting employee records, are implemented within this module.

Database Module:

This module, named db, encapsulates functionalities related to database operations using SQLite.It provides a Database class that serves as an interface for performing CRUD operations on an SQLite database file.Operations such as inserting new records, updating existing ones, deleting records, and fetching all records are defined within this module.

2. Survey of Technologies

2.1 **Software Description**

The software for the Employee Management System is developed using Python, which is known for its simplicity and efficiency. The graphical user interface is created using Tkinter and CustomTkinter, libraries that are popular for developing desktop applications in Python. MySQL is employed as the database management system to store and manage data related to movies and ticket availability. Together, these technologies create a robust and user-friendly application for booking movie tickets.

2.2 **Languages**

2.2.1 **SQL**

Structured Query Language (SQL) is a standard programming language used for managing and manipulating relational databases. In this project, SQL is used extensively to perform various database operations such as creating tables, inserting data, updating records, and fetching data from the database. The primary SQL operations involved in the project are:

Create Tables: SQL commands are used to define the structure of the database, creating tables to store employee information.

Insert Data: SQL INSERT statements are used to add employee details into the database.

Update Data: SQL UPDATE statements are used to modify the employee details with ease in usability .
.

Delete Data: SQL SELECT statements are used to delete the respective employee as per need .

2.2.2 **Python**

Python is the primary programming language used to develop the Employee Management System. It is widely known for its readability and ease of use, making it an ideal choice for both novice and experienced developers. Python's extensive library support allows for rapid development and integration of various functionalities.

Key Python components and libraries used in the project include:

Tkinter and CustomTkinter: These libraries are used to create the graphical user interface (GUI) of the application. Tkinter is the standard GUI toolkit for Python, while CustomTkinter provides additional customization options for creating modern and visually appealing interfaces.

MySQL Connector: This is a Python library that facilitates communication between Python and MySQL databases. It allows the execution of SQL queries from within the Python code, enabling seamless database operations.

File Handling: Python's built-in file handling capabilities are used to generate and save text files containing booking details, providing users with a tangible record of their transactions.

3. REQUIREMENT ANALYSIS

3.1 Requirement Analysis

Functional Requirements:

User Interface:

The system should have a graphical user interface (GUI) created using Tkinter.The GUI should include an input form for entering employee details and a table for displaying employee records.

Data Validation:

The system should validate input fields to ensure all necessary details are provided before adding or updating records.Appropriate error messages should be displayed if required fields are missing or if data formats are incorrect.

Database Interaction:

The system should interact with an SQLite database to store, retrieve, update, and delete employee records.A separate module (`db.py`) should handle all database operations.

Error Handling and Notifications:

The system should handle errors gracefully, providing meaningful error messages to the user.Successful operations (e.g., adding, updating, deleting records) should be acknowledged with confirmation messages.

Accessibility and Usability:

The GUI should be user-friendly, with clearly labeled input fields and buttons.The application should support basic keyboard and mouse interactions for ease of use.

Data Persistence:

Employee records should be persistently stored in the SQLite database, ensuring data is retained across application sessions.

Non-Functional Requirements :

These requirements define the overall characteristics and qualities of the system:

Usability:

The system should be intuitive and easy to learn for users with varying levels of technical expertise.The interface should be well-organized and user-friendly, promoting efficient data entry and retrieval.

Performance:

The system should respond quickly to user actions and data manipulation.It should be able to handle a reasonable number of concurrent users without significant performance degradation.

Reliability:

The system should be reliable and function consistently with minimal downtime.Data should be protected from loss or corruption through backups and error handling.

Scalability:

The system should be adaptable to accommodate future growth in the number of employees or data volume.

Security:

The system should implement security measures to protect sensitive employee information from unauthorized access or modification.User access should be controlled with different permission levels.

Maintainability:

The system should be well-documented and easy to maintain, allowing for future modifications and bug fixes.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

HARDWARE SPECIFICATION

PROCESSOR : INTEL i3

MEMORY SIZE : 4GB

HDD : 256GB

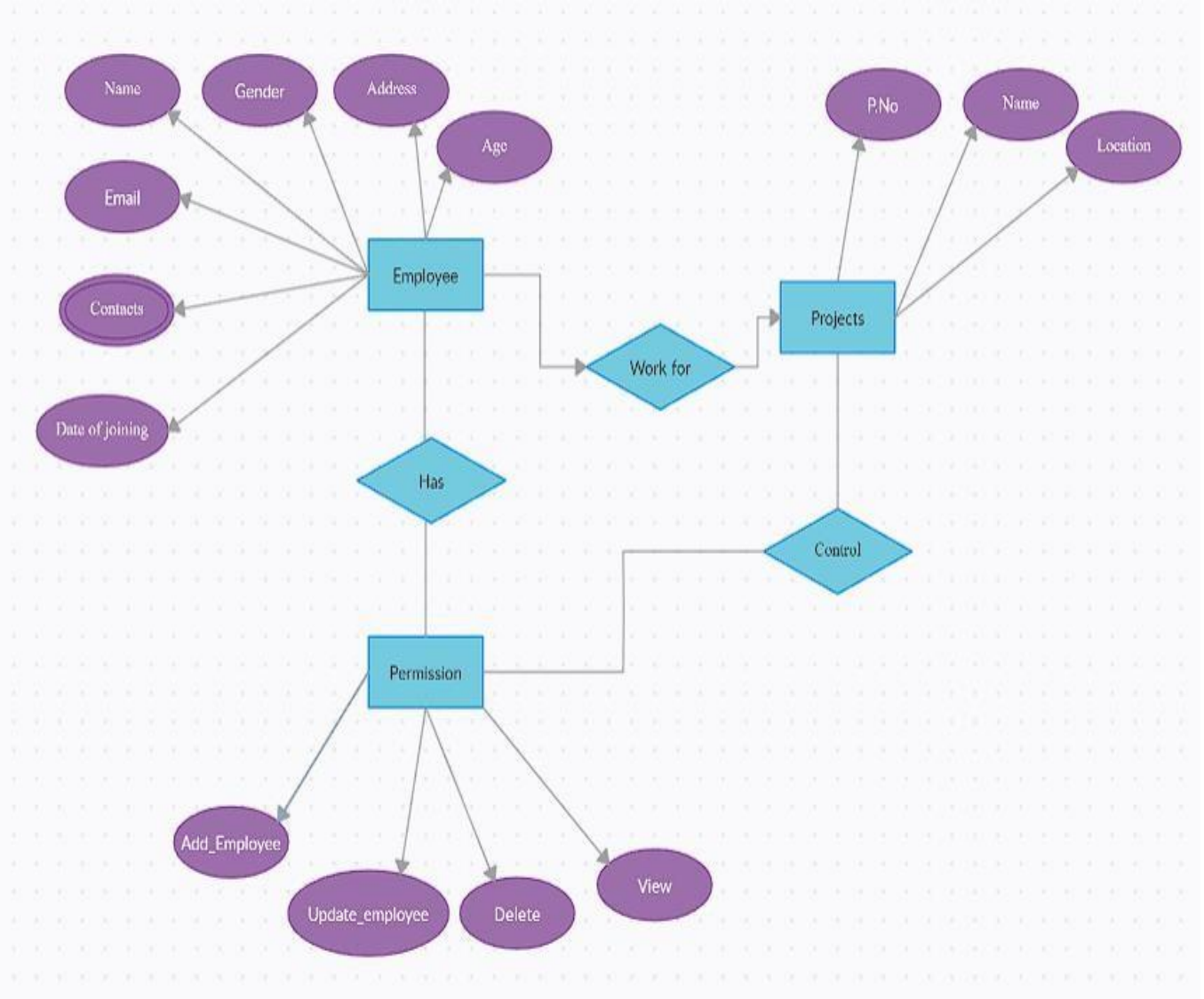
SOFTWARE SPECTFICATION

OPERATING SYSTEM : WINDOWS 11

GUI INTERFACE : PYTHON

BACKEND : MY SQL

3.3 **ERDIAGRAM**



4 PROGRAM CODE :

//MAIN.PY

```
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="#2c3e50")
root.state("zoomed")

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

# Entries Frame
entries_frame = Frame(root, bg="#2D3142")
entries_frame.pack(side=TOP, fill=X)
title = Label(entries_frame, text="Employee Management System", font=("Calibri", 18, "bold"), bg="#2D3142", fg="white")
title.grid(row=0, columnspan=2, padx=10, pady=20, sticky="w")

lblName = Label(entries_frame, text="Name", font=("calibri", 17,"bold"),bg='#2D3142', 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)
txtName.grid(row=1, column=1, padx=10, pady=10, sticky="w")
```

```
lblAge = Label(entries_frame, text="Age", font=("calibri", 17,"bold"),bg='#2D3142', 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)
txtAge.grid(row=1, column=3, padx=10, pady=10, sticky="w")

lbldoj = Label(entries_frame, text="D.O.J", font=("calibri", 17,"bold"),bg='#2D3142', 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)
txtDoj.grid(row=2, column=1, padx=10, pady=10, sticky="w")

lblEmail = Label(entries_frame, text="Email", font=("calibri", 17,"bold"),bg='#2D3142', 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)
txtEmail.grid(row=2, column=3, padx=10, pady=10, sticky="w")

lblGender = Label(entries_frame, text="Gender", font=("calibri", 17,"bold"),bg='#2D3142', 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", 17,"bold"),bg='#2D3142', 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)
txtContact.grid(row=3, column=3, padx=10, sticky="w")

lblAddress = Label(entries_frame, text="Address", font=("calibri", 17,"bold"),bg='#2D3142', 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))
```

```
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='#2D3142')
```

```
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='green', bd=0).grid(row=0, column=0,padx=10)

btnEdit = Button(btn_frame, command=update_employee, text="Update Details", width=15, font=("Calibri", 16, "bold"),

        fg="white", bg="#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",

        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",

        bd=0).grid(row=0, column=3, padx=10)


# Table Frame

tree_frame = Frame(root, bg="#ecf0f1")

tree_frame.place(x=0, y=480, width=1980, height=520)

style = ttk.Style()

style.configure("mystyle.Treeview", font=('Calibri', 18),

        rowheight=50)  # 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()

entries_frame = Frame(root, bg="#2D3142")
entries_frame.pack(side=TOP, fill=X)
title = Label(entries_frame, text="Employee Management System", font=("Calibri", 18, "bold"), bg="#2D3142", fg="white")
title.grid(row=0, columnspan=2, padx=10, pady=20, sticky="w")


lblName = Label(entries_frame, text="Name", font=("calibri", 17,"bold"),bg='#2D3142', 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)
txtName.grid(row=1, column=1, padx=10, pady=10, sticky="w")
```

```
lblAge = Label(entries_frame, text="Age", font=("calibri", 17,"bold"),bg='#2D3142', 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)
txtAge.grid(row=1, column=3, padx=10, pady=10, sticky="w")

lbldoj = Label(entries_frame, text="D.O.J", font=("calibri", 17,"bold"),bg='#2D3142', 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)
txtDoj.grid(row=2, column=1, padx=10, pady=10, sticky="w")

lblEmail = Label(entries_frame, text="Email", font=("calibri", 17,"bold"),bg='#2D3142', 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)
txtEmail.grid(row=2, column=3, padx=10, pady=10, sticky="w")

lblGender = Label(entries_frame, text="Gender", font=("calibri", 17,"bold"),bg='#2D3142', 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", 17,"bold"),bg='#2D3142', 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)
txtContact.grid(row=3, column=3, padx=10, sticky="w")

lblAddress = Label(entries_frame, text="Address", font=("calibri", 17,"bold"),bg='#2D3142', 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))
```

```
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='#2D3142')
```

```
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='green', bd=0).grid(row=0, column=0,padx=10)

btnEdit = Button(btn_frame, command=update_employee, text="Update Details", width=15, font=("Calibri", 16, "bold"),

        fg="white", bg="#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",

        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",

        bd=0).grid(row=0, column=3, padx=10)


# Table Frame

tree_frame = Frame(root, bg="#ecf0f1")

tree_frame.place(x=0, y=480, width=1980, height=520)

style = ttk.Style()

style.configure("mystyle.Treeview", font=('Calibri', 18),

        rowheight=50)  # 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()
```

//DB.PY

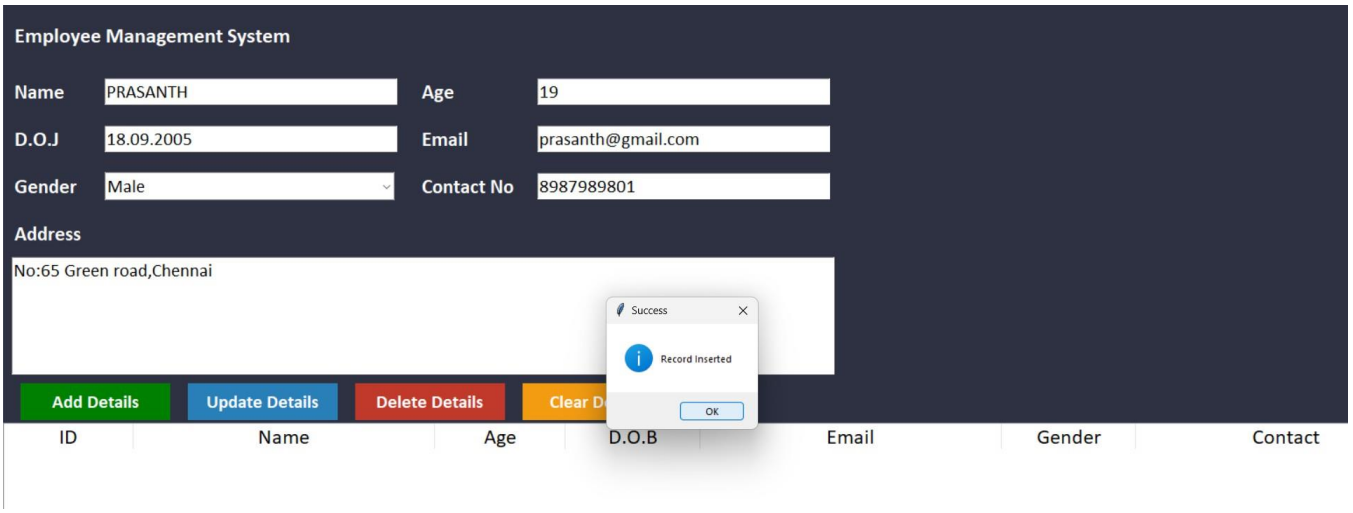
```
import sqlite3


class Database:
    def __init__(self, db):
        self.con = sqlite3.connect
self.cur = self.con.cursor() sql
=CREATE TABLE IF NOT
EXISTS employees(

            id Integer Primary Key,
            name text,
            age text,
            doj text,
            email text,
            gender text,
            contact text,
            address text
        )
        """
        self.cur.execute(sql)
        self.con.commit()

    # Insert Function
    def insert(self, name, age, doj, email, gender, contact, address):
        self.cur.execute("insert into employees values (NULL,?,?,?,?,?,?,?)",
                         (name, age, doj, email, gender, contact, address))
        self.con.commit()

    # Fetch All Data from DB
    def fetch(self):
        self.cur.execute("SELECT * from employees")
        rows = self.cur.fetchall()
        # print(rows)
        return rows
```

```
# Delete a Record in DB
def remove(self, id):
    self.cur.execute("delete from employees where id=?", (id,))
    self.con.commit()



# Insert Function
def insert(self, name, age, doj, email, gender, contact, address):
    self.cur.execute("insert into employees values (NULL,?,?,?,?,?,?,?)",
                     (name, age, doj, email, gender, contact, address))
    self.con.commit()



# Fetch All Data from DB
def fetch(self):
    self.cur.execute("SELECT * from employees")
    rows = self.cur.fetchall()
    # print(rows)
    return rows
```

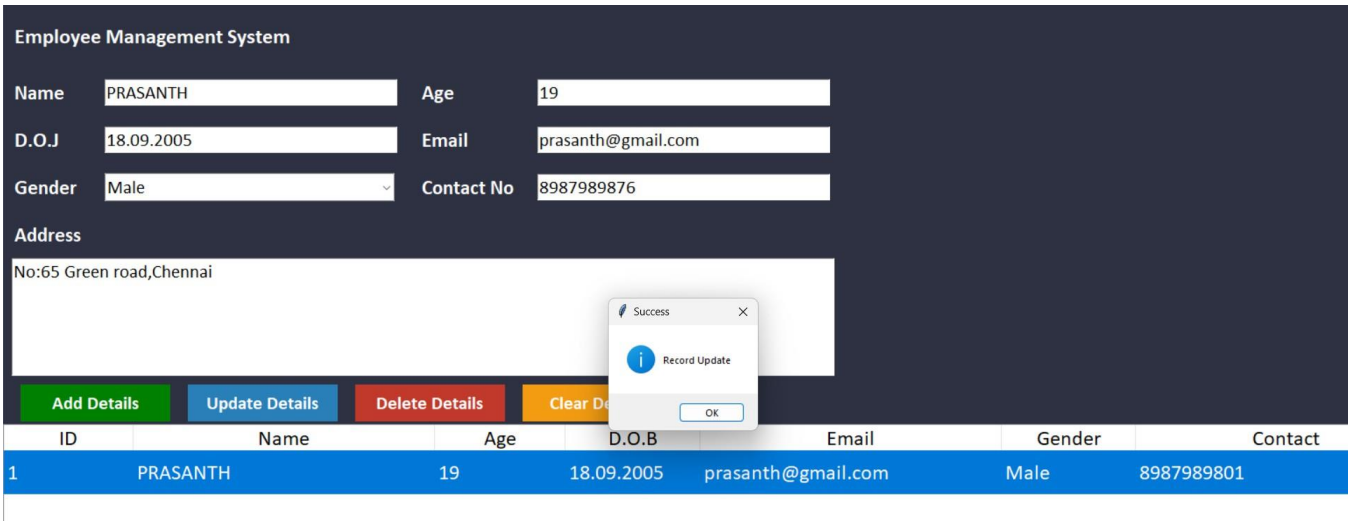
```
    # Update a Record in DB
    def update(self, id, name, age, doj, email, gender, contact, address):
        self.cur.execute(
            "update employees set name=?, age=?, doj=?, email=?, gender=?, contact=?, address=? where id=?",
            (name, age, doj, email, gender, contact, address, id))
        self.con.commit()
```

4.1 RESULT:

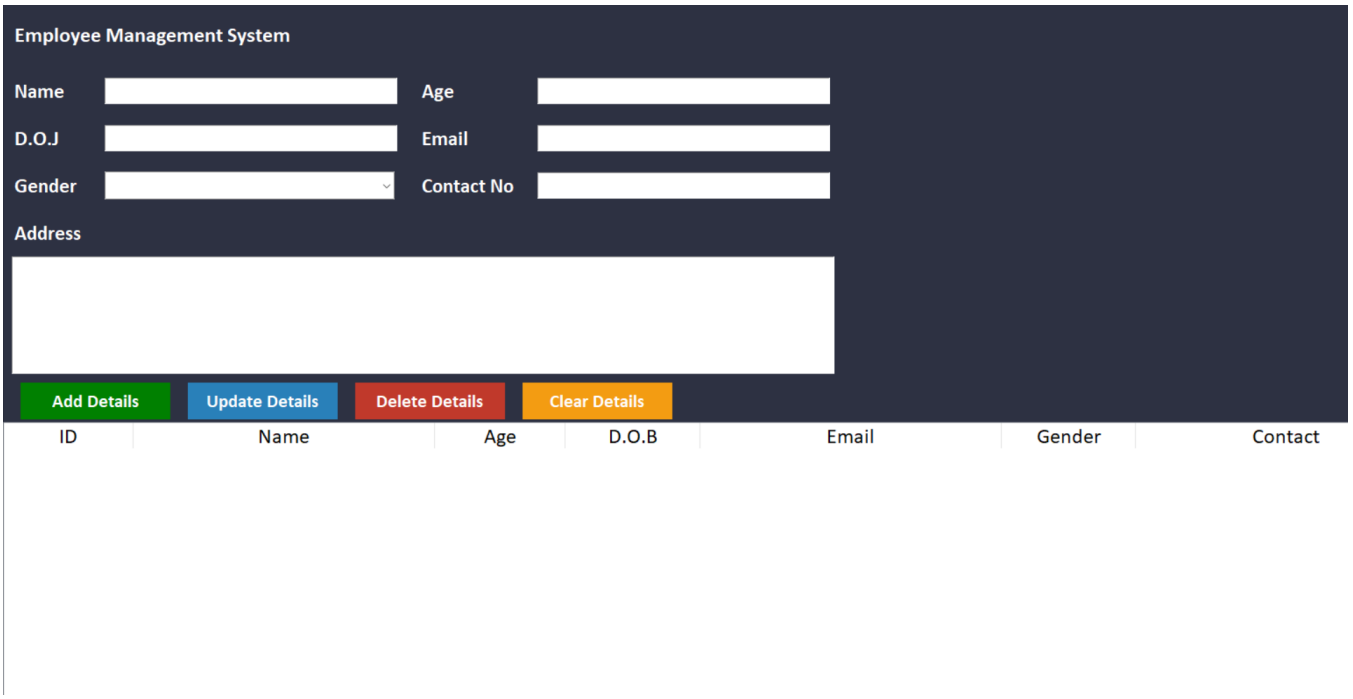
1. Add Employee:



2. Update Employee:



3. Delete Employee:



4.2 CONCLUSION:

In conclusion, this Python-based Employee Management System, built with Tkinter for the GUI and leveraging a relational database (SQL) for data storage, demonstrates the potential for streamlined HR processes. While this example focuses on CRUD (Create, Read, Update, Delete) functionalities, it lays the groundwork for future development. By expanding on this foundation, organizations can integrate features like Advanced search and filtering for employee data. ,Reporting and analytics dashboards for deeper insights , Integration with other HR systems for a unified platform. Through its intuitive interface and robust functionality, the system enables efficient handling of employee data, from personal details to performance evaluations. By automating routine tasks and ensuring data integrity, the EMS enhances productivity while minimizing errors. Its flexibility, scalability, and integration capabilities empower organizations to adapt to evolving needs seamlessly. Ultimately, the EMS emerges as a cornerstone for fostering organizational efficiency, transparency, and growth in today's dynamic business landscape.

REFERENCES

ADD LINKS ,BOOKS,REFERED TO DEVELOP THIS PROJECT

REFERENCES

1 **Manish Kumar Srivastava, A.K Tiwari, "A Study of Behavior of Maruti SX4 and Honda City Custo Jaipur", Pacific Business Review- Quarterly Referred Journal, Zenith International Journal of Multi disciplinary Research Vol.4, Issue 4, pp. 77-90, Apr2011.**

1 **M.Prasanna Mohan Raj, Jishnu Sasikumar, S.Sriram , "A Study of Customers Brand Preference in SUVS and MUVS: Effect on Marketing Mix Variables", International Referred Research Journal Vol.- IV, Issue-1, pp. 48-58, Jan2013.**

2 **Nikhil Monga, Bhuvender Chaudhary, "Car Market and Buying behavior - study on Consumer Perception", IJRMEC Vol.2, Issue-2, pp. 44-63, Feb2012.**