EMPLOYEE MANAGEMENT SYSTEM

Submitted in partial fulfillment of the requirements of the
Second Year of Engineering in Computer Science & Engineering

By

Dhamankar Soham Santosh
Roll No.10

Kapse Swaroop Ajinkya
Roll No.21

Patil Neha Dnyaneshwar
Roll No.36

Raut Rohit Ramchandra
Roll No.41

Guide:

Prof. S. M. PATIL

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

KONKAN GYANPEETH COLLEGE OF ENGINEERING

UNIVERSITY OF MUMBAI

(2023-2024)

CERTIFICATE

This is to certify the project entitled **"Employee Management System"** is a bonafide work of **"Dhamankar Soham Santosh (10), Kapse Swaroop Ajinkya (21), Patil Neha Dnyaneshwar (36), Raut Rohit Ramchandra (41)"** submitted to be University of Mumbai in partial fulfillment of the requirement for the award of the **"S.E."** in **"Computer Science & Engineering".**

(Prof. S. M. Patil)
Guide

(Prof. S. M. Patil)
Head of Department

(Dr. V. J. Pillewan)
Principal

KONKAN GYANPEETH COLLEGE OF ENGINEERING, KARJAT

(Affiliated to University of Mumbai, Approved by A.I.C.T.E., New Delhi.)

Konkan Gyanpeeth Shaikshanik Sankul, Vengaon Road, Dahivali, Karjat, Dist.-Raigad-410201. (M.S.)

Department of Computer Science & Engineering

Project Report Approval

This is project report entitled **"Employee Management System"** by work of **Dhamankar Soham Santosh (10), Kapse Swaroop Ajinkya (21), Patil Neha Dnyaneshwar (36), Raut Rohit Ramchandra (41)** is approved for the full completion of S.E. of Department **of Computer Science & Engineering**.

Examiners

1.....

2.....

Date:

Place:

DECLARATION

We declare that this written submission represents our ideas in our own words and where other ideas or words have been included, we have adequately excited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Dhamankar Soham Santosh (10) _____

Kapse Swaroop Ajinkya (21) _____

Patil Neha Dnyaneshwar (36) _____

Raut Rohit Ramchandra (41) _____

Date:

ACKNOWLEDGEMENT

We would like to express our gratitude and appreciation to all those who gave us the possibility to complete this project and this report.

We thanks to our project in charge & guide Prof. S. M. Patil, without whose guidance this project would not have become successful a great source of inspiration to us who help to stimulating suggestions and encouragement helped us a lot during our project work.

Finally, we would like to thank our parents and mentors for the support they gave us throughout the course.

INDEX

ABSTRACT

In modern organizations, managing employees efficiently is crucial for maintaining productivity and fostering a positive work environment. With the advancement of technology, Employee Management Systems (EMS) have become indispensable tools for organizations and companies. Employee Management System Project is designed to keep track of employee information in any company. It stores data such as their employee's personal information and salary details. Employee Management System Python project assists in the automation of manual operations, saving both time and money. This system protects the professional and personal information of employees and the company.

This abstract presents an overview of an efficient EMS developed using Python programming language and MySQL database which streamlines various tasks and enhances organizational effectiveness. To build an Employee Management System project in Python with source code we have to connect the MySQL database to our program.

The proposed EMS is designed to automate essential functions such as Add, Update, Delete, Clear, Search, Reset, Fetch, etc., employee record. Leveraging Python's versatility and extensive libraries, the system offers a user-friendly interface for both administrators and employees, ensuring ease of use and accessibility.

❖ Lists of Figures:

CHAPTER NO 1

INTRODUCTION

INTRODUCTION

❖ Introduction to Python:

Python is a high-level, versatile, and interpreted programming language known for its simplicity and readability.

Created by Guido van Rossum and first released in 1991, Python has gained widespread popularity and has become one of the most popular programming languages globally.

- The main features of the Python language include:
  1. Simple and clean syntax enhances code readability
  2. Easy to start
  3. Widely used for developing Desktop and Web Application
  4. Extensive Standard Library
  5. Cross-platform compatibility

Overall, Python's simplicity, versatility, extensive library ecosystem, and strong community support make it an ideal choice for developing mini-projects across various domains and applications.

❖ Introduction to MySQL:

MySQL database is a relational database managed by the MySQL Server, which is a popular open-source relational database management system (RDBMS). MySQL databases are structured collections of data organized in tables, where each table consists of rows and columns.

A MySQL database is a relational database managed by the MySQL Server, which is a popular open-source relational database management system (RDBMS). MySQL databases are structured collections of data organized in tables, where each table consists of rows and columns.

❖ V S Code:

Visual Studio Code for the Web provides a free, zero-install Microsoft Visual Studio Code experience running entirely in your browser, allowing you to quickly and safely browse source code repositories and make lightweight code changes.

VS Code for the Web provides a browser-based experience for navigating files and repositories and committing lightweight code changes. However, if you need access to a runtime to run, build, or debug your code, you want to use platform features such as a terminal, or you want to run extensions that aren't supported in the web, we recommend moving your work to the desktop application, GitHub for the full capabilities of VS Code. In addition, VS Code Desktop lets you use a full set of keyboard shortcuts not limited by your browser.

❖ Introduction to Employee Management System:

Employee Management System is a distributed application, developed to maintain the details of employees working in any organization. It maintains the information about the personal details of their employee. The application is developed using Python programming language and MySQL database. It is simple to understand and can be used by anyone who is not even familiar with simple employee system.

The purpose of an employee management system is to help improve workforce productivity, identify ways to engage and retain talent, and alleviate administrative burdens for HR professionals. Achieving greater efficiency through the use of technology can also help control costs and minimize compliance risks.

An Employee Management System project typically involves requirements gathering, system design, development, testing, deployment, and maintenance phases. It requires collaboration between HR personnel, IT professionals, and end-users to ensure that the system meets the organization's needs effectively. Employee management system is developed to manage the data and information of an employee in a company. It is developed to override the problems prevailing in the practicing manual system.

❖ **Existing System:**

The project eliminates the paperwork, human faults, manual delay and speeds up the process, which have helped in making the organizations work easier and more efficient.

❖ **Problem Statement:**

To design and develop the Employee management system (application) using Python to streamline the process of managing employee information within an organization. The system should provide functionalities to efficiently handle various tasks related to employee data management.

❖ **Objectives:**

- This project aims to simplify the task maintaining records of the employees of company.
- To develop a well-designed database to store employee information.
- To add, update and delete the records of an employee.
- To simplify the process of modifying the data of employee in organization.
- To minimize the time of organization.

❖ **Proposed System:**

- The proposed system provides detailed general information about the employee along the department and salary details.
- It enhances the human resource management in adding, viewing and updating employee's details and generates various reports regarding employee's skill and experience.

❖ **Scope:**

- A great system will allow you to classify your employees based on department, location, and designation for easy data management
- The system that you select should come with an employee database management platform that allows you to centralize HR data for every department, region, and business entity.
- With this feature, you can manually add employee details in just a few steps.
- Employees can also access and edit their personal information to ensure error-free records.

CHAPTER NO 2

REQUIREMENT ANALYSIS

REQUIREMENT ANALYSIS

1. **Software Requirements:**

- Visual Studio Code
- Python
- MySQL Workbench
- Windows Operating System

2. **Hardware Requirements:**

- Processor: Intel core i5
- PC with 8GB RAM and 515 SSD

3. **System Requirements:**

❖ **Functional Requirements:**

The Modules description of Employee Management System project. These modules will be developed in PYTHON source code and MYSQL database.

1. **Employee Records Management:**

The system should enable users to add, save, update, delete, clear and search employee records efficiently.

2. **Data Maintenance:**

Ensure the system maintains accurate and up-to-date employee data.

3. **User-Friendly Interface:**

Design the system with a user-friendly interface for easy navigation and operation.

4. **View Employee Details:**

Employee will be able to view their own details.

❖ **Non-Functional Requirements:**

1. **Security:**

Implement security measures to protect employee data and system integrity.

2. **Usability:**

Ensure the system is easy to use and understand for all users.

3. **Reliability:**

The system should be reliable, providing consistent performance and accurate data.

4. **Maintainability:**

Ensure the system is easy to maintain and update as needed.

❖ **Pre-Processor Segment:**

- from tkinter import*:

Imports all classes, functions, and constants from the tkinter module without prefixing them.

- from tkinter import ttk:

Imports the ttk submodule from the tkinter module, which provides themed widget classes.

- from PIL import Image,ImageTk

Imports the Image and ImageTk modules from the Python Imaging Library (PIL), used for handling images.

- import mysql.connector

Imports the mysql.connector module, which is used to connect to MySQL databases.

- from tkinter import messagebox

Imports the messagebox submodule from the tkinter module, which is used to display message boxes.

❖ **Functions:**

- Add Data: It is used to create a function to add data to the database. This function should get the data from the entry boxes, connect to the database, execute an SQL insert statement, and then close the database connection.
- Fetch Data: It is used to create a function to fetch data from the database. This function should connect to the database, execute an SQL select statement, fetch the data, and then close the database connection. The fetched data should be displayed in the GUI.
- Update Data: It is used to create a function to update data in the database. This function should get the updated data from the entry boxes, connect to the database, execute an SQL update statement, and then close the database connection.
- Delete Data: It is used to create a function to delete data from the database. This function should get the ID proof of the data to be deleted from the entry boxes, connect to the database, execute an SQL delete statement, and then close the database connection.
- Clear Data: It is used to create the function to clear data from the database. This function should refer to clearing all the information and settings stored within the system.
- Reset Data: It is used to create a function to reset the data in the entry boxes to their default values.
- Search Data: It is used to create a function to search for data in the database based on the specified search criteria. This function should get the search criteria from the entry boxes, connect to the database, execute an SQL select statement with a where clause, fetch the data, and then close the database connection. The searched data should be displayed in the GUI.
- Get Cursor: It is used to create a function to get the selected row from the table and display the data in the entry boxes.
- Show All: It is used to show all data from database.

❖ **Libraries:**

- Tkinter: It is a standard Python interface to the Tk GUI toolkit. It is used to create the graphical user interface (GUI) of the application.
- PIL (Pillow): It is a library in Python that adds support for opening, manipulating, and saving many different images file formats. It is used to display images in the GUI.
- ttk: It is a module in Tkinter that provides a set of widgets that are more modern and customizable than the standard Tkinter widgets. It is used to create the GUI elements, such as labels, entry boxes, and buttons.
- StringVar: It is a variable class in Tkinter that is used to store string values. It is used to store the data entered in the entry boxes.
- Combobox: It is a widget in ttk that allows the user to select a value from a drop-down list. It is used to select the department and ID proof type in the GUI.
- Button: It is a widget in Tkinter that is used to trigger an action when clicked. It is used to submit, update, delete, and search for data in the database.
- Entry: It is a widget in Tkinter that is used to enter and display a single line of text. It is used to enter the employee data in the GUI.
- Label: It is a widget in Tkinter that is used to display text or images. It is used to display the labels in the GUI.
- Frame: It is a container widget in Tkinter that is used to group other widgets together. It is used to group the GUI elements in the main window.
- Scrollbar: It is a widget in Tkinter that is used to scroll the content of a window. It is used to scroll the content of the table in the GUI.
- Treeview: It is a widget in ttk that is used to display data in a table format. It is used to display the employee data in the GUI.
- messagebox: It is a module in Tkinter that is used to display messages and dialog boxes. It is used to display messages and dialog boxes in the GUI.

- SQL: SQL (Structured Query Language) is a standard language for managing and manipulating relational databases. It is used to insert, update, delete, and search for data in the MySQL database.

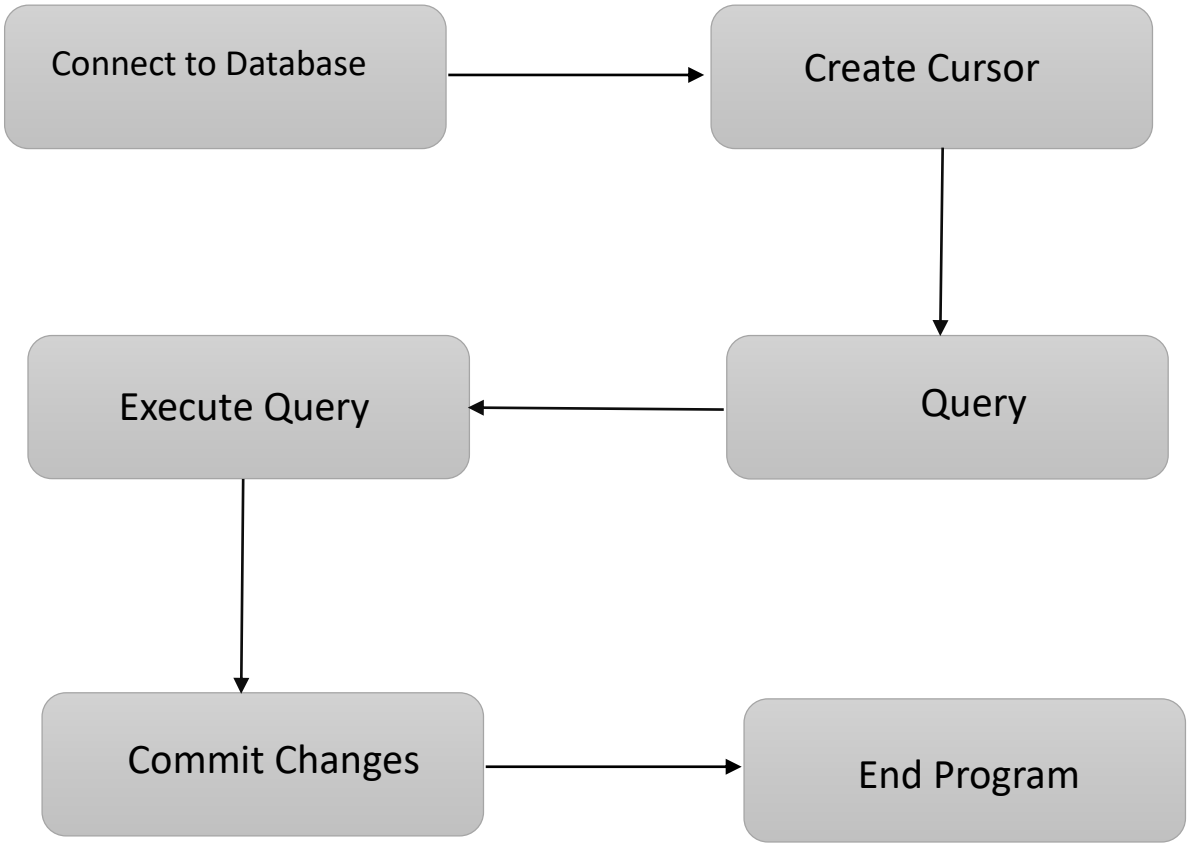
❖ **Methods:**

- _init_: This is a special method in Python classes, which is called when an object is created from the class. It is used to initialize the attributes of the class.
- create_widgets: This method is used to create the GUI elements, such as labels, entry boxes, and buttons.
- add_data: This method is used to insert data into the employee_management table in the database.
- fetch_data: This method is used to fetch data from the employee_management table in the database and display it in the GUI.
- get_cursor: This method is used to get the selected row from the table and display the data in the entry boxes.
- update_data: This method is used to update the data in the employee_management table in the database.
- delete_data: This method is used to delete data from the employee_management table in the database.
- reset_data: This method is used to reset the data in the entry boxes to their default values.
- clear_data: This method is used to clear or erase all the stored information or data within the system.
- search_data: This method is used to search for data in the employee_management table in the database based on the specified search criteria.
- Show All: This method is used to show all data of Employee from Database.

CHAPTER NO 3

SYSTEM DESIGN

❖ System Connectivity:



For MySQL, we use the mysql-connector-python library to connect to the database. Firstly, install the library:

- pip install mysql-connector-python

Connect to the MySQL database:

- import mysql.connector

Connect to MySQL database:

- conn=mysql.connector.connect(host="localhost",username="root",password="ROH IT8082206879",database="employee")

Create a cursor object:

- my_cursor = conn.cursor()

Execute SQL query:

- my_cursor.execute("SELECT * FROM table_name")

Fetch data

- my_cursor.execute(“Select * from employee_management “)
- data = my_cursor.fetchall()

#Add data

- conn=mysql.connector.connect(host="localhost",username="root",password="ROH IT8082206879",database="employee")
    - my_cursor=conn.cursor()
    - my_cursor.execute(“ Insert employee_management values(?,?,?,?,?,?,?,?.?,?,?,?,?,?)”,(
      self.var_dep.get(),
      self.var_name.get(),
      self.var_designition.get(),
      self.var_email.get(),
      self.var_address.get(),
      self.var_married.get(),
      self.var_dob.get(),
      self.var_doj.get(),
      self.var_idproofcombo.get(),
      self.var_gender.get(),
      self.var_phone.get(),
      self.var_country.get(),
      self.var_salary.get(),
      self.var_idproof.get()
      ))

Update data

- conn=mysql.connector.connect(host="localhost",username="root",password="ROH IT8082206879",database="employee")
    - my_cursor=conn.cursor()

- my_cursor.execute('update employee_management set Department=%s,Name=%s,Designition=%s,Email=%s,Address=%s,Married_Status=%s,DOB=%s,DOJ=%s,ID_Proof_Type=%s,Gender=%s,Phone_No=%s,Country=%s,Salary=%s where ID_Proof=%s', (
  self.var_dep.get(),
  self.var_name.get(),
  self.var_designition.get(),
  self.var_email.get(),
  self.var_address.get(),
  self.var_married.get(),
  self.var_dob.get(),
  self.var_doj.get(),
  self.var_idproofcombo.get(),
  self.var_gender.get(),
  self.var_phone.get(),
  self.var_country.get(),
  self.var_salary.get(),
  self.var_idproof.get()
  ))

#Delete data

- sql='delete from employee_management where ID_Proof=%s'

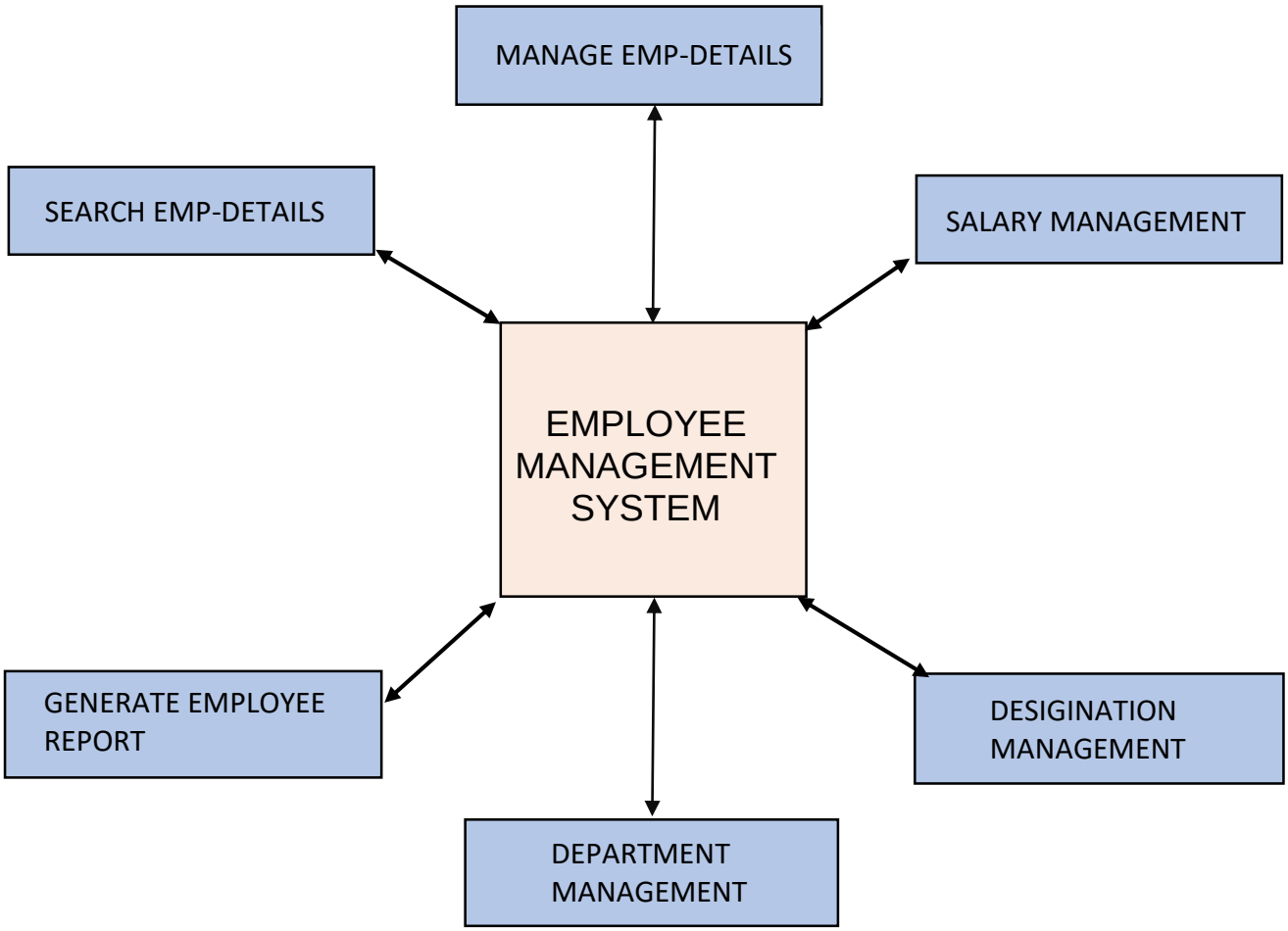
value=(self.var_idproof.get(),)

my_cursor.execute(sql,value)

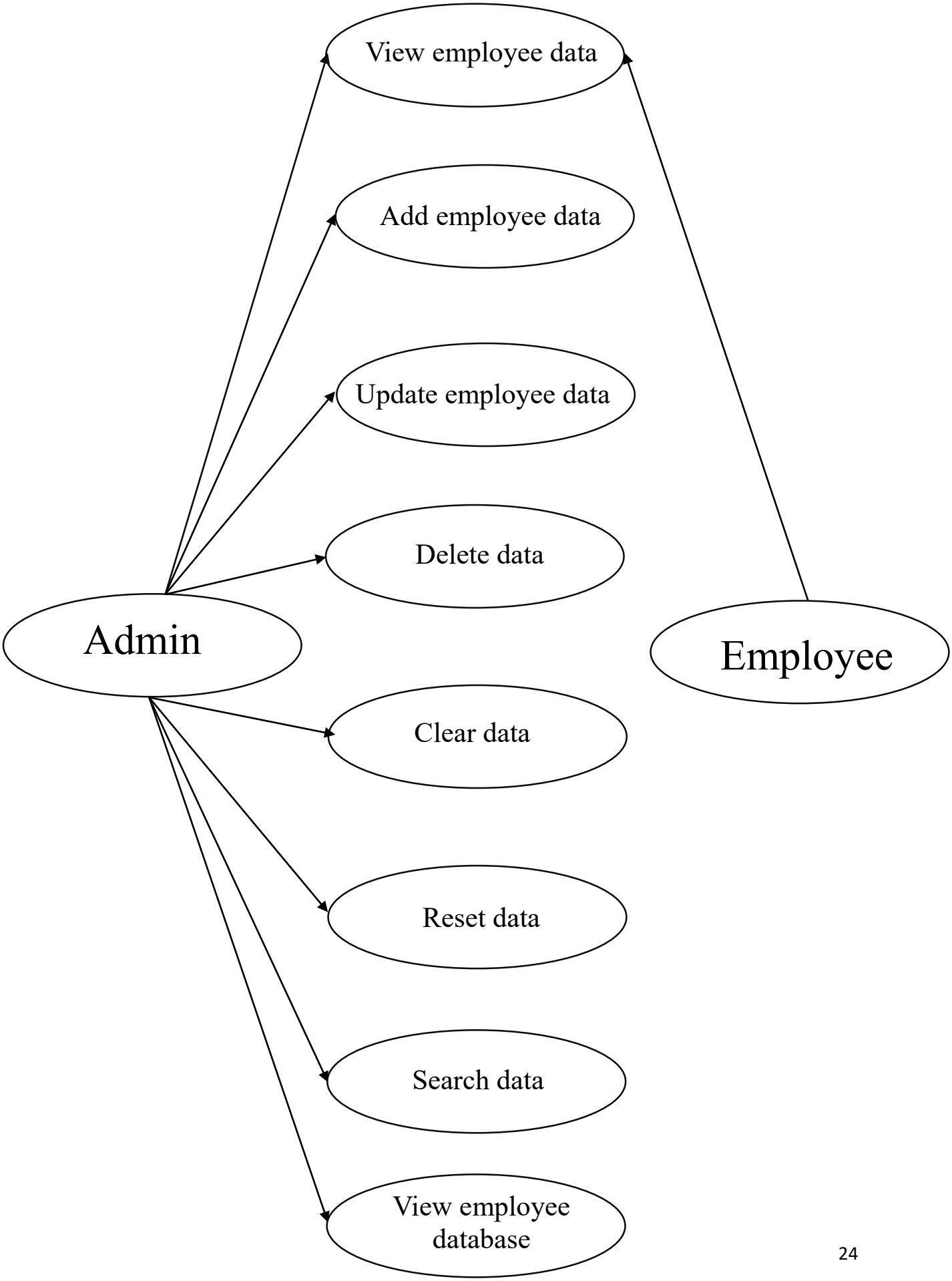
Close cursor and connection

- cursor.close()
- conn.commit()
- conn.close(

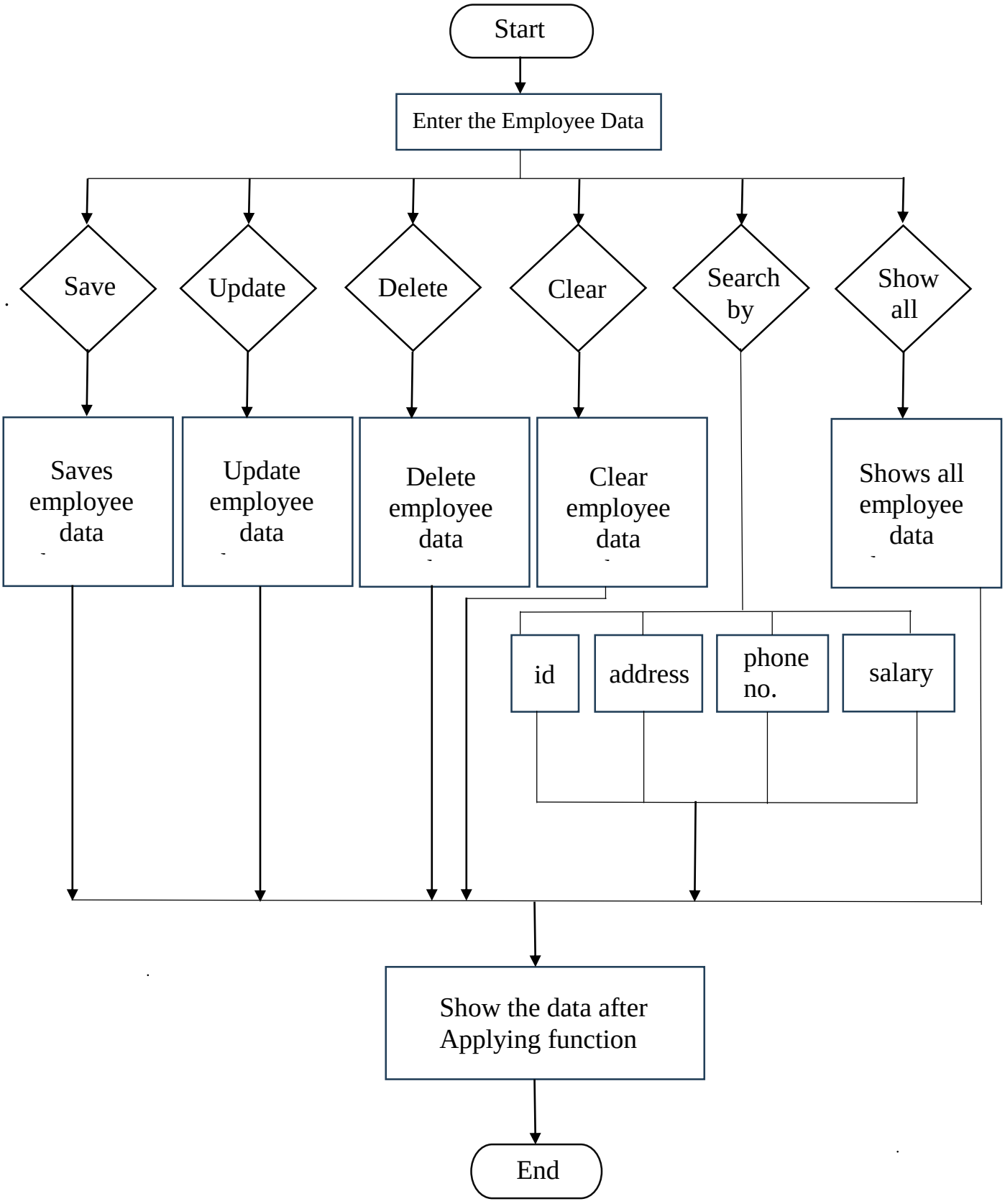
❖ Data Flow diagram :



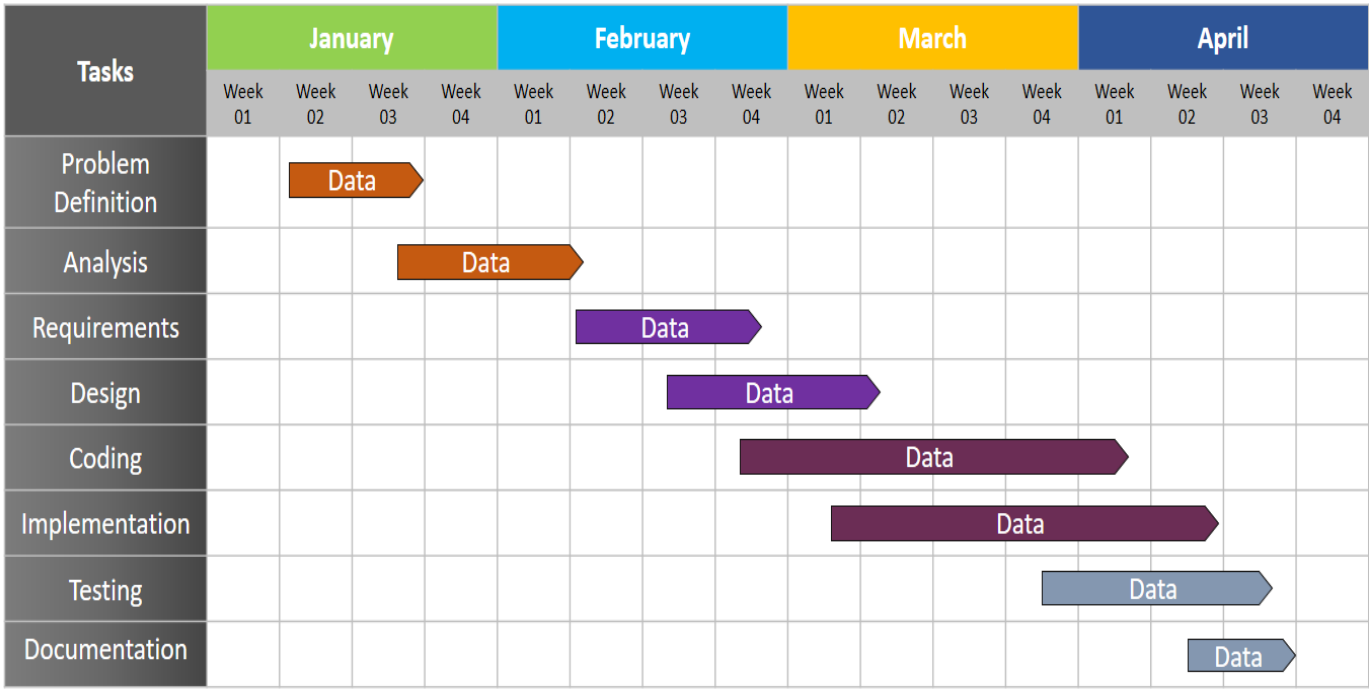
❖ **Use Case Diagram:**



❖ Flow Chart :


Start
Enter the Employee Data
Save
Update
Delete
Clear
Search by
Show all
Saves employee data
Update employee data
Delete employee data
Clear employee data
Shows all employee data
id
address
phone no.
salary
Show the data after Applying function
End

❖ Gantt Chart :


Tasks
January
February
March
April
Week 01
Week 02
Week 03
Week 04
Problem Definition
Analysis
Requirements
Design
Coding
Implementation
Testing
Documentation
Data

CHAPTER NO 4

IMPLEMENTATION

IMPLEMENTATION

```
from tkinter import*
from tkinter import ttk
from PIL import Image,ImageTk
import mysql.connector
from tkinter import messagebox
class Employee:
    def __init__(self,root):
        self.root=root
        self.root.geometry("1366x767+0+0")
        self.root.title("Employee Management System")
        self.var_dep=StringVar()
        self.var_name=StringVar()
        self.var_designition=StringVar()
        self.var_email=StringVar()
        self.var_address=StringVar()
        self.var_married=StringVar()
        self.var_dob=StringVar()
        self.var_doj=StringVar()
        self.var_idproofcombo=StringVar()
        self.var_idproof=StringVar()
        self.var_gender=StringVar()
        self.var_phone=StringVar()
        self.var_country=StringVar()
        self.var_salary=StringVar()
        #Title
```

```
    lbl_title=Label(self.root,text="EMPLOYEE MANAGEMENT
SYSTEM",font=("times new
roman",35,"bold"),foreground="black",background="Bisque")
        lbl_title.place(x=0,y=0,width=1370,height=50)


        #Logo
        img_logo=Image.open("Emp_Img/Emp_logo.jpg")
        img_logo=img_logo.resize((50,50))
        self.photo_logo=ImageTk.PhotoImage(img_logo)
        self.logo=Label(self.root,image=self.photo_logo,background="Bisque")
        self.logo.place(x=210,y=0,width=50,height=50)


        #Frame for placing images
        img_frame=Frame(self.root,border=2,relief=RIDGE,background="white")
        img_frame.place(x=0,y=50,width=1370,height=130)


        # 1 Image
        img_1=Image.open("Emp_Img/Emp_Img_1.png")
        img_1=img_1.resize((455,160))
        self.photo_1=ImageTk.PhotoImage(img_1)
        self.img1=Label(img_frame,image=self.photo_1)
        self.img1.place(x=0,y=0,width=455,height=160)
        # 2 Image
        img_2=Image.open("Emp_Img/Emp_Img_2.png")
        img_2=img_2.resize((476,160))
        self.photo_2=ImageTk.PhotoImage(img_2)
        self.img2=Label(img_frame,image=self.photo_2)
```

```
    self.img2.place(x=455,y=0,width=476,height=160)
      # 3 Image
      img_3=Image.open("Emp_Img/Emp_Img_4.png")
      img_3=img_3.resize((457,160))
      self.photo_3=ImageTk.PhotoImage(img_3)
      self.img3=Label(img_frame,image=self.photo_3)
      self.img3.place(x=931,y=0,width=457,height=160)

     #Main frame
      Main_frame=Frame(self.root,border=2,relief=RIDGE,background="Lavender")
      Main_frame.place(x=5,y=190,width=1345,height=500)

    #UpperFrame
upper_frame=LabelFrame(Main_frame,border=2,relief=RIDGE,background="Lavender
",text="EmployeeInformation",font=("timesnewroman",10,"bold"),foreground="black")
upper_frame.place(x=10,y=0,width=1320,height=235)

  #Labels and Entry Fields
   Label_dep=Label(upper_frame,text="Department
:",font=("arial",10,"bold"),foreground="black",background="Lavender")
   Label_dep.grid(row=0,column=0,padx=2,sticky=W)

combo_dep=ttk.Combobox(upper_frame,textvariable=self.var_dep,font=("arial",10,"bol
d"),width=17,state="readonly"
 combo_dep["value"]=("Select Department","Administration","Security","Financial
Accounting","Markting","Sales","Project Management","Maintenance Department")
combo_dep.current(0)
 combo_dep.grid(row=0,column=1,padx=2,pady=10,sticky=W)
```

```
#Name

Label_name=Label(upper_frame,text="Name
:",font=("arial",10,"bold"),foreground="black",background="Lavender")
Label_name.grid(row=0,column=2,padx=22,pady=7,sticky=W)


Text_name=ttk.Entry(upper_frame,textvariable=self.var_name,font=("arial",10,"bold"),
width=20)
 Text_name.grid(row=0,column=3,padx=0,pady=7)


#Designition

Label_designition=Label(upper_frame,text="Designition
:",font=("arial",10,"bold"),foreground="black",background="Lavender")
Label_designition.grid(row=1,column=0,padx=2,pady=7,sticky=W)


Text_designition=ttk.Entry(upper_frame,textvariable=self.var_designition,font=("arial",
10,"bold"),width=20)
Text_designition.grid(row=1,column=1,padx=2,pady=7,sticky=W)


#Email

 Label_email=Label(upper_frame,text="Email
:",font=("arial",10,"bold"),foreground="black",background="Lavender")
  Label_email.grid(row=1,column=2,padx=22,pady=7,sticky=W)


Text_email=ttk.Entry(upper_frame,textvariable=self.var_email,font=("arial",10,"bold"),
width=20)
  Text_email.grid(row=1,column=3,padx=2,pady=7,sticky=W)


 #Address
```

```
 Label_address=Label(upper_frame,text="Address
:",font=("arial",10,"bold"),foreground="black",background="Lavender")
 Label_address.grid(row=2,column=0,padx=2,pady=7,sticky=W)

Text_address=ttk.Entry(upper_frame,textvariable=self.var_address,font=("arial",10,"bol
d"),width=20)
 Text_address.grid(row=2,column=1,padx=2,pady=7,sticky=W)

 #Married
 Label_married=Label(upper_frame,text="Married Status
:",font=("arial",10,"bold"),foreground="black",background="Lavender")
    Label_married.grid(row=2,column=2,padx=2,sticky=W)

combo_married=ttk.Combobox(upper_frame,textvariable=self.var_married,font=("arial
",10,"bold"),width=17,state="readonly")
        combo_married["value"]=("Select Status","Married","Unmarried")
        combo_married.current(0)
        combo_married.grid(row=2,column=3,padx=2,pady=10,sticky=W)

#Phone
 Label_phone=Label(upper_frame,text="Phone No.
:",font=("arial",10,"bold"),foreground="black",background="Lavender")
   Label_phone.grid(row=0,column=4,padx=22,pady=7,sticky=W)

Text_phone=ttk.Entry(upper_frame,textvariable=self.var_phone,font=("arial",10,"bold"
),width=20)
 Text_phone.grid(row=0,column=5,padx=2,pady=7,sticky=W)
```

```
 #Dob
 Label_dob=Label(upper_frame,text="DOB
:",font=("arial",10,"bold"),foreground="black",background="Lavender")
    Label_dob.grid(row=0,column=6,padx=22,pady=7,sticky=W)

Text_dob=ttk.Entry(upper_frame,textvariable=self.var_dob,font=("arial",10,"bold"),wid
th=20)
 Text_dob.grid(row=0,column=7,padx=2,pady=7,sticky=W)

 #Country
  Label_country=Label(upper_frame,text="Country
:",font=("arial",10,"bold"),foreground="black",background="Lavender")
   Label_country.grid(row=1,column=4,padx=22,pady=7,sticky=W)

Text_country=ttk.Entry(upper_frame,textvariable=self.var_country,font=("arial",10,"bo
ld"),width=20)
 Text_country.grid(row=1,column=5,padx=2,pady=7,sticky=W)

 #Date of joining
 Label_doj=Label(upper_frame,text="DOJ
:",font=("arial",10,"bold"),foreground="black",background="Lavender")
  Label_doj.grid(row=1,column=6,padx=22,pady=7,sticky=W)

Text_doj=ttk.Entry(upper_frame,textvariable=self.var_doj,font=("arial",10,"bold"),widt
h=20)
 Text_doj.grid(row=1,column=7,padx=2,pady=7,sticky=W)

#Gender
```

```
Label_gender=Label(upper_frame,text="Gender
:",font=("arial",10,"bold"),foreground="black",background="Lavender")
Label_gender.grid(row=2,column=4,padx=22,sticky=W)
combo_gender=ttk.Combobox(upper_frame,textvariable=self.var_gender,font=("arial",
10,"bold"),width=17,state="readonly")
combo_gender["value"]=("Select Gender","Male","Female","Transgender")
combo_gender.current(0)
combo_gender.grid(row=2,column=5,padx=2,pady=10,sticky=W)

#ID Proof
combo_id=ttk.Combobox(upper_frame,textvariable=self.var_idproofcombo,font=("arial
",10,"bold"),width=14,state="readonly")
combo_id["value"]=("Select Id Proof","Aadhaar Card","Pan Card","Voter
ID","Employee ID","Driving License")
combo_id.current(0)
combo_id.grid(row=2,column=6,padx=22,pady=10,sticky=W)

Text_id=ttk.Entry(upper_frame,textvariable=self.var_idproof,font=("arial",10,"bold"),w
idth=20)
Text_id.grid(row=2,column=7,padx=2,pady=7,sticky=W)

#Salary
Label_salary=Label(upper_frame,text="Salary
:",font=("arial",10,"bold"),foreground="black",background="Lavender")
Label_salary.grid(row=3,column=0,padx=2,pady=7,sticky=W)

Text_salary=ttk.Entry(upper_frame,textvariable=self.var_salary,font=("arial",10,"bold")
,width=20)
Text_salary.grid(row=3,column=1,padx=2,pady=7,sticky=W)
```

```
#Button
btn_add=Button(upper_frame,text="SAVE",command=self.add_data,font=("arial",10,"
bold"),foreground="black",background="gray")
btn_add.place(x=80,y=170,width=100,height=40)

btn_update=Button(upper_frame,text="UPDATE",command=self.update_data,font=("a
rial",10,"bold"),foreground="black",background="gray")
btn_update.place(x=380,y=170,width=100,height=40)

btn_delete=Button(upper_frame,text="DELETE",command=self.delete_data,font=("ari
al",10,"bold"),foreground="black",background="gray")
btn_delete.place(x=680,y=170,width=100,height=40)

btn_clear=Button(upper_frame,text="CLEAR",command=self.reset_data,font=("arial",
10,"bold"),foreground="black",background="gray")
btn_clear.place(x=980,y=170,width=100,height=40)

#Lower Frame
lower_frame=LabelFrame(Main_frame,border=2,relief=RIDGE,background="Lavender
",text="Employee Information Table",font=("times new
roman",10,"bold"),foreground="black")
lower_frame.place(x=10,y=240,width=1320,height=248)

 #Search Frame
search_frame=LabelFrame(lower_frame,border=2,relief=RIDGE,background="Lavend
er",text=" Search Employee Information",font=("times new
roman",10,"bold"),foreground="black")
 search_frame.place(x=5,y=0,width=1305,height=60)
```

```
 search_by=Label(search_frame,border=2,relief=RIDGE,background="gray",text="
Search By :",width=12,font=("times new roman",11,"bold"),foreground="black")
search_by.grid(row=0,column=0,sticky=W,padx=5,pady=5)


 #Search
  self.var_com_search=StringVar()

combo_txt_search=ttk.Combobox(search_frame,textvariable=self.var_com_search,font
=("arial",10,"bold"),width=14,state="readonly")
combo_txt_search["value"]=("Select Option","Phone No.","ID_Proof",)
combo_txt_search.current(0)
combo_txt_search.grid(row=0,column=1,padx=22,pady=10,sticky=W)
 self.var_search=StringVar()

Text_search=ttk.Entry(search_frame,textvariable=self.var_search,font=("arial",10,"bold
"),width=20)
 Text_search.grid(row=0,column=2,padx=2,pady=7,sticky=W)

btn_search=Button(search_frame,text="Search",command=self.search_data,font=("arial
",10,"bold"),width=15,foreground="black",background="gray")
 btn_search.grid(row=0,column=3,sticky=W,padx=25)

 btn_showall=Button(search_frame,text="Show
All",command=self.fetch_data,font=("arial",10,"bold"),width=15,foreground="black",b
ackground="gray"
btn_showall.grid(row=0,column=4,sticky=W,padx=25)


      #Table frame
```

```
 table_frame=Frame(lower_frame,border=2,relief=RIDGE,background="Black")
        table_frame.place(x=5,y=65,width=1305,height=158)

        #Scrollbar
        Scroll_x=ttk.Scrollbar(table_frame,orient=HORIZONTAL)
        Scroll_y=ttk.Scrollbar(table_frame,orient=VERTICAL)

self.employee_table=ttk.Treeview(table_frame,columns=("dep","name","degi","email",
"address","married","dob","doj","idproofcombo","id
proof","gender","phone","country","salary"),xscrollcommand=Scroll_x.set,yscrollcom
mand=Scroll_y.set)
        Scroll_x.pack(side=BOTTOM,fill=X)
        Scroll_y.pack(side=RIGHT,fill=Y)
        Scroll_x.config(command=self.employee_table.xview)
        Scroll_y.config(command=self.employee_table.yview)

        self.employee_table.heading("dep",text="Department")
        self.employee_table.heading("name",text="Name")
        self.employee_table.heading("degi",text="Designition")
        self.employee_table.heading("email",text="Email")
        self.employee_table.heading("address",text="Address")
        self.employee_table.heading("married",text="Married Status")
        self.employee_table.heading("dob",text="DOB")
        self.employee_table.heading("doj",text="DOJ")
        self.employee_table.heading("idproofcombo",text="ID Type")
        self.employee_table.heading("id proof",text="ID Proof")
        self.employee_table.heading("gender",text="Gender")
```

```
self.employee_table.heading("phone",text="Phone No")
    self.employee_table.heading("country",text="Country")
    self.employee_table.heading("salary",text="Salary")

    self.employee_table["show"]="headings"

    self.employee_table.column("dep",width=100)
    self.employee_table.column("name",width=100)
    self.employee_table.column("degi",width=100)
    self.employee_table.column("email",width=100)
    self.employee_table.column("address",width=100)
    self.employee_table.column("married",width=100)
    self.employee_table.column("dob",width=100)
    self.employee_table.column("doj",width=100)
    self.employee_table.column("idproofcombo",width=100)
    self.employee_table.column("id proof",width=100)
    self.employee_table.column("gender",width=100)
    self.employee_table.column("phone",width=100)
    self.employee_table.column("country",width=100)
    self.employee_table.column("salary",width=100)

    self.employee_table.pack(fill=BOTH,expand=1)
    self.employee_table.bind("<ButtonRelease>",self.get_cursor)
   self.fetch_data()

#**************function Decleration***************
def add_data(self):
```

```
    if self.var_dep.get()=="" or self.var_email.get()=="":
            messagebox.showerror("Error","All fields are required")
        else:
            try:
conn=mysql.connector.connect(host="localhost",username="root",password="ROHIT8
082206879",database="employee")
  my_cursor=conn.cursor()
   my_cursor.execute("insert into employee_management
values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",(

self.var_dep.get(),
self.var_name.get(),
self.var_designition.get(),
self.var_email.get(),
self.var_address.get(),
self.var_married.get(),
self.var_dob.get(),
self.var_doj.get(),
self.var_idproofcombo.get(),
self.var_idproof.get(),
self.var_gender.get(),
self.var_phone.get(),
self.var_country.get(),
self.var_salary.get()
))
  conn.commit()
  self.fetch_data()
    conn.close()
     messagebox.showinfo("Success","Employee has been added!",parent=self.root)
  except Exception as es:
```

```
  messagebox.showerror("Error",f"Due To:{str(es)}",parent=self.root)


#Fetch Data
def fetch_data(self):
conn=mysql.connector.connect(host="localhost",username="root",password="ROHIT8
082206879",database="employee")
        my_cursor=conn.cursor()
        my_cursor.execute("select * from employee_management")
        data=my_cursor.fetchall()
        if len(data)!=0:
            self.employee_table.delete(*self.employee_table.get_children())
         for i in data:
               self.employee_table.insert("",END,values=i)
         conn.commit()
          conn.close()


    # Get Cursor
    def get_cursor(self,event=""):
        cursor_row=self.employee_table.focus()
        content=self.employee_table.item(cursor_row)
        data=content['values']


        self.var_dep.set(data[0])
        self.var_name.set(data[1])
        self.var_designition.set(data[2])
        self.var_email.set(data[3])
        self.var_address.set(data[4])
```

```
  self.var_married.set(data[5])
        self.var_dob.set(data[6])
        self.var_doj.set(data[7])
        self.var_idproofcombo.set(data[8])
        self.var_idproof.set(data[9])
        self.var_gender.set(data[10])
        self.var_phone.set(data[11])
        self.var_country.set(data[12])
        self.var_salary.set(data[13])

    def update_data(self):
        if self.var_dep.get()=="" or self.var_email.get()=="":
            messagebox.showerror("Error","All fields are required")
        else:
            try:
                update=messagebox.askyesno('Update','Are you sure to update this employee
data')
                if update>0:

conn=mysql.connector.connect(host="localhost",username="root",password="ROHIT8
082206879",database="employee")
                    my_cursor=conn.cursor()
                    my_cursor.execute('update employee_management set
Department=%s,Name=%s,Designition=%s,Email=%s,Address=%s,Married_Status=%
s,DOB=%s,DOJ=%s,ID_Proof_Type=%s,Gender=%s,Phone_No=%s,Country=%s,Sala
ry=%s where ID_Proof=%s',(

self.var_dep.get(),
```

```
self.var_name.get(),
self.var_designition.get(),
self.var_email.get(),
self.var_address.get(),
self.var_married.get(),
self.var_dob.get(),
self.var_doj.get(),
self.var_idproofcombo.get(),
self.var_gender.get(),
self.var_phone.get(),
self.var_country.get(),
self.var_salary.get(),
self.var_idproof.get()
))
                else:
                    if not update:
                        return
                conn.commit()
                self.fetch_data()
                conn.close()
            messagebox.showinfo("Success","Employee Successfully
Updated",parent=self.root)
            except Exception as es:
                messagebox.showerror("Error",f"Due To:{str(es)}",parent=self.root)


    # Delete Function
    def delete_data(self):
        if self.var_idproof.get()=="":
            messagebox.showerror("Error","All fields are required")
        else:
```

```
 try:
            Delete=messagebox.askyesno('Delete','Are you sure to delete this employee
data',parent=self.root)
            if Delete>0:

conn=mysql.connector.connect(host="localhost",username="root",password="ROHIT8
082206879",database="employee")
               my_cursor=conn.cursor()
               sql='delete from employee_management where ID_Proof=%s'
               value=(self.var_idproof.get(),)
               my_cursor.execute(sql,value)
            else:
               if not Delete:
                  return
            conn.commit()
            self.fetch_data()
            conn.close()
            messagebox.showinfo("Delete","Employee Successfully
Deleted",parent=self.root)
         except Exception as es:
            messagebox.showerror("Error",f"Due To:{str(es)}",parent=self.root)


   #reset
   def reset_data(self):
      self.var_dep.set("Select Department")
      self.var_name.set("")
      self.var_designition.set("")
```

```
  self.var_email.set("")
        self.var_address.set("")
        self.var_married.set("Select Status")
        self.var_dob.set("")
        self.var_doj.set("")
        self.var_idproofcombo.set("Select Id Proof")
        self.var_idproof.set("")
        self.var_gender.set("Select Gender")
        self.var_phone.set("")
        self.var_country.set("")
        self.var_salary.set("")


    #Search
    def search_data(self):
        if self.var_com_search.get()==" or self.var_search.get()==":
            messagebox.showerror('Error','Please select option')
        else:
            try:
conn=mysql.connector.connect(host="localhost",username="root",password="ROHIT8
082206879",database="employee")
                my_cursor=conn.cursor()
                my_cursor.execute('SELECT * FROM employee_management WHERE ' +
str(self.var_com_search.get()) + ' LIKE "%" + str(self.var_search.get()) + "%"')
                rows=my_cursor.fetchall()
                if len(rows)!=0:
                    self.employee_table.delete(*self.employee_table.get_children())
                    for i in rows:
```

```
  self.employee_table.insert("",END,values=i)
            conn.commit()
            conn.close()
           except Exception as es:
              messagebox.showerror("Error",f"Due To:{str(es)}",parent=self.root)


if __name__=="__main__":
    root=Tk()
    obj=Employee(root)
root.mainloop()
```

CHAPTER NO 5

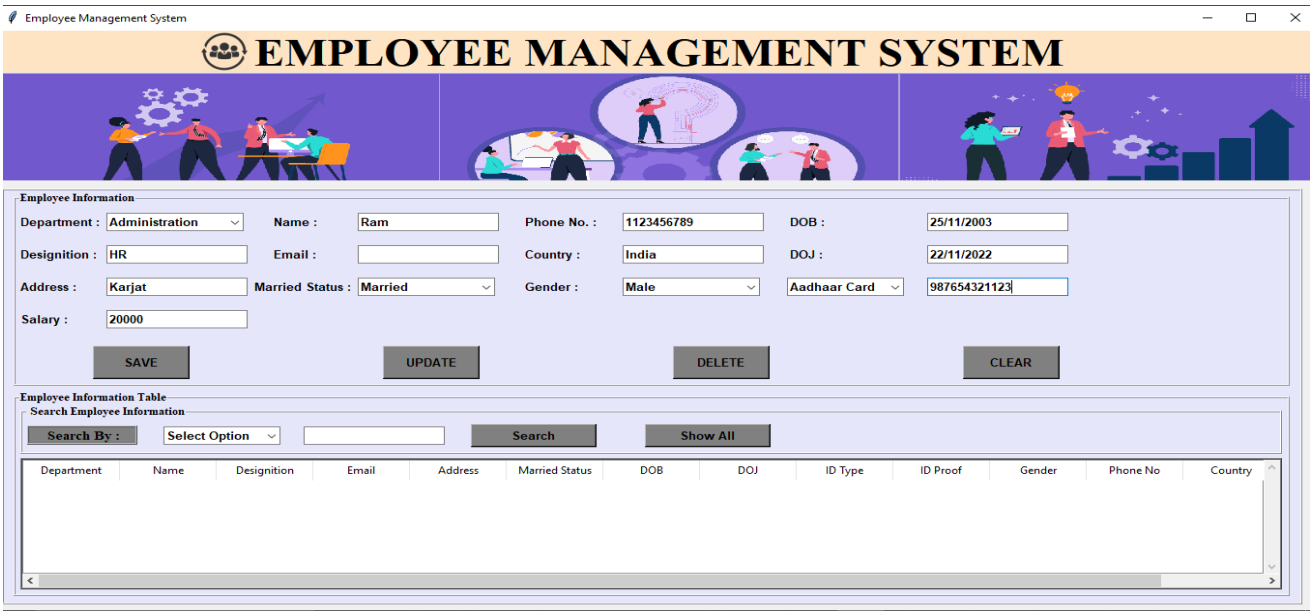
TESTING

TESTING

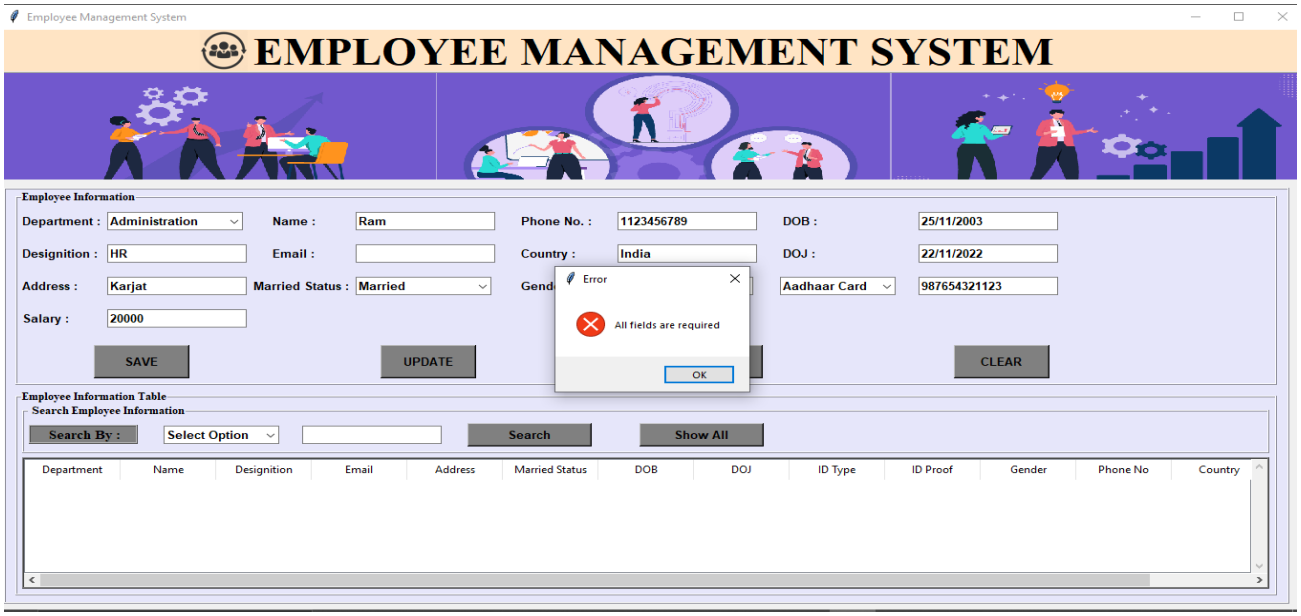
This section provides a visual presentation of all test-samples have been performed upon the program's functionality.

1. Saving data into Database:

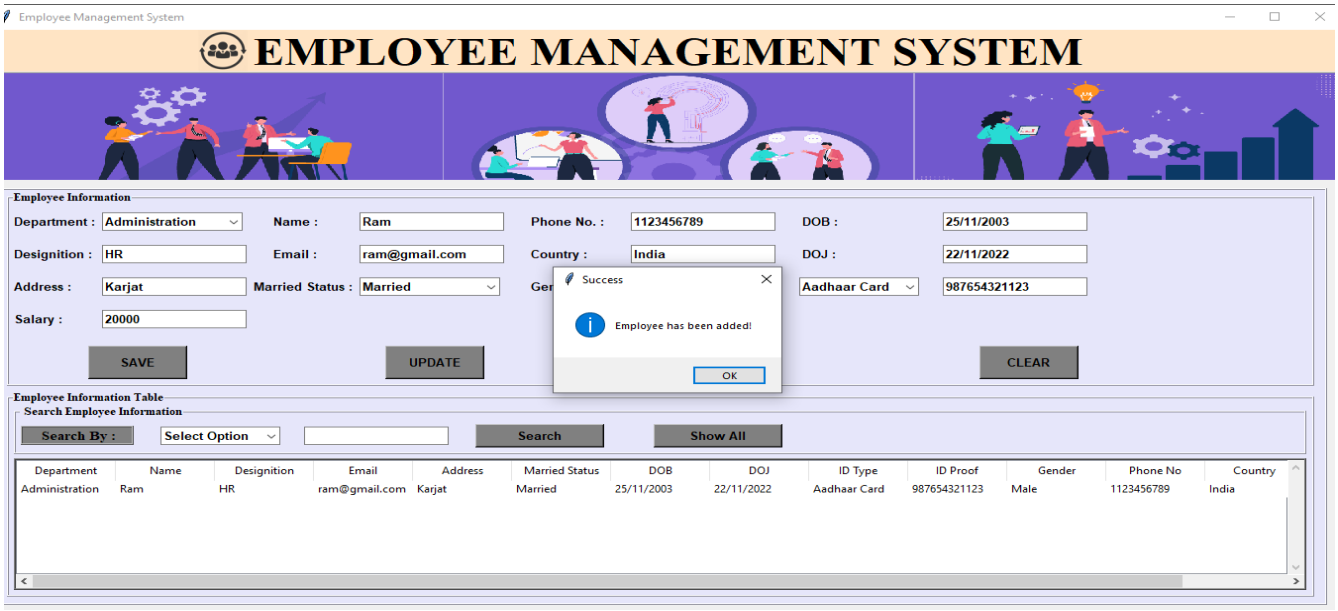
The first test involves adding a new employee's data to the database. For simplicity, only data for the parent table is going to be input, thus the system will throw a warning, but after clicking on the "OK" button, the data will be successfully saved into the database.



After clicking on the "Save" button, the check-result could be seen

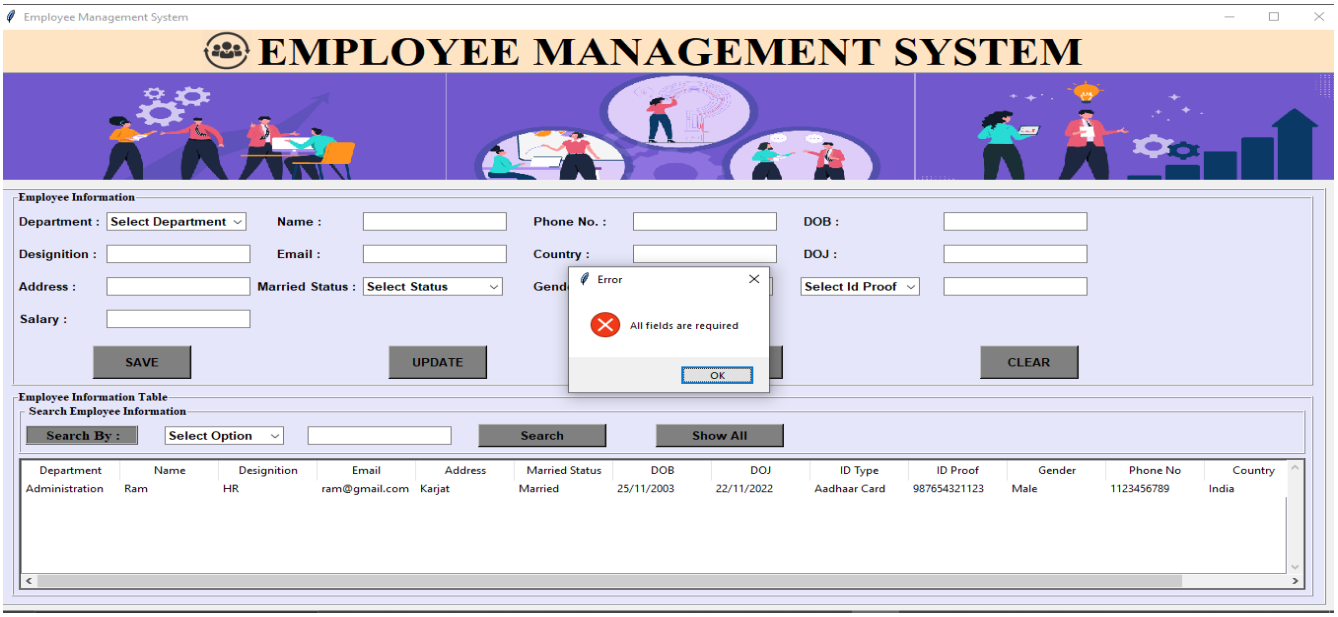


The message box, in the figure above, says All fields are required so we solve this error by filling all the data in entry box.



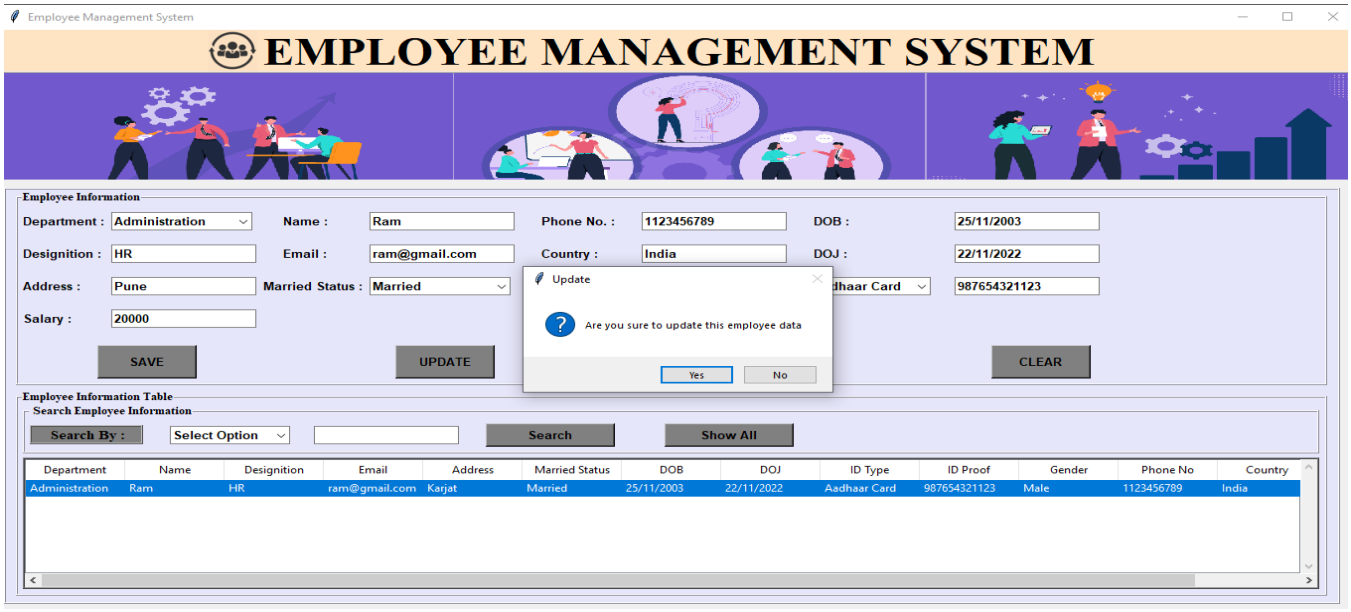
In this way, we can add records into the database.

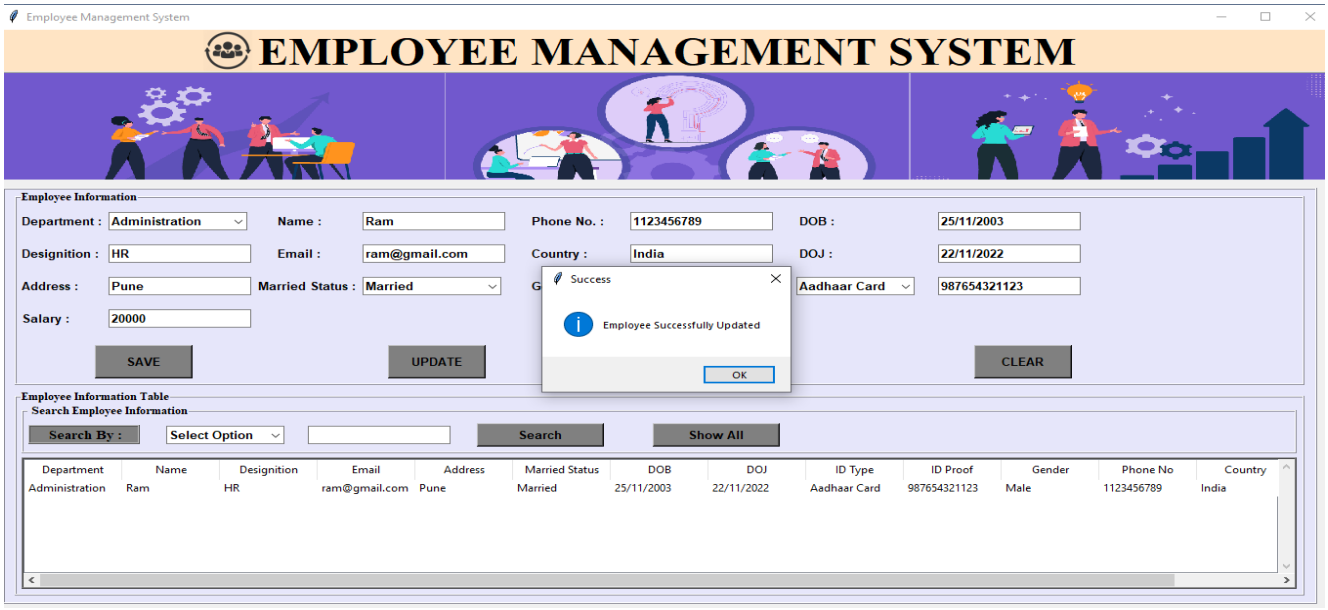
2. Updating the data into Database:



A Warning Message is displayed on the screen, as we have not the sufficient information to perform such kind of operation!

We can change the address to “Pune” and click the update button afterwards, The result is supposed to be: address as “Pune” instead of the previous one “Karjat”:

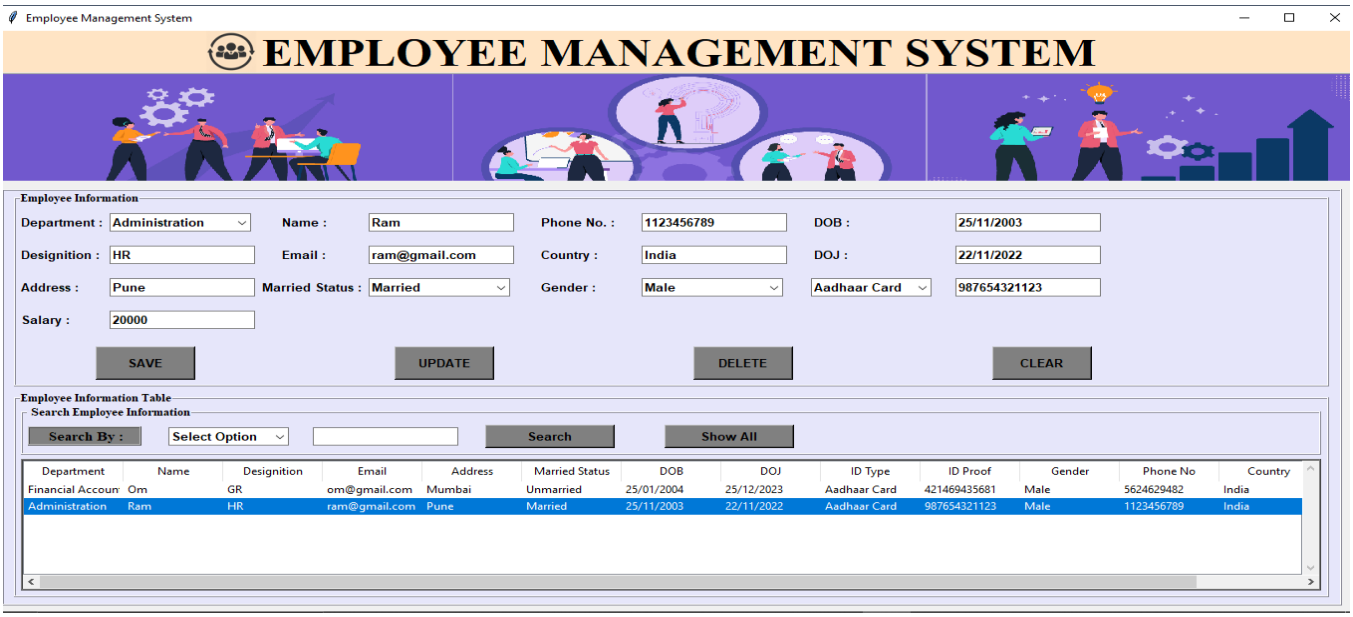


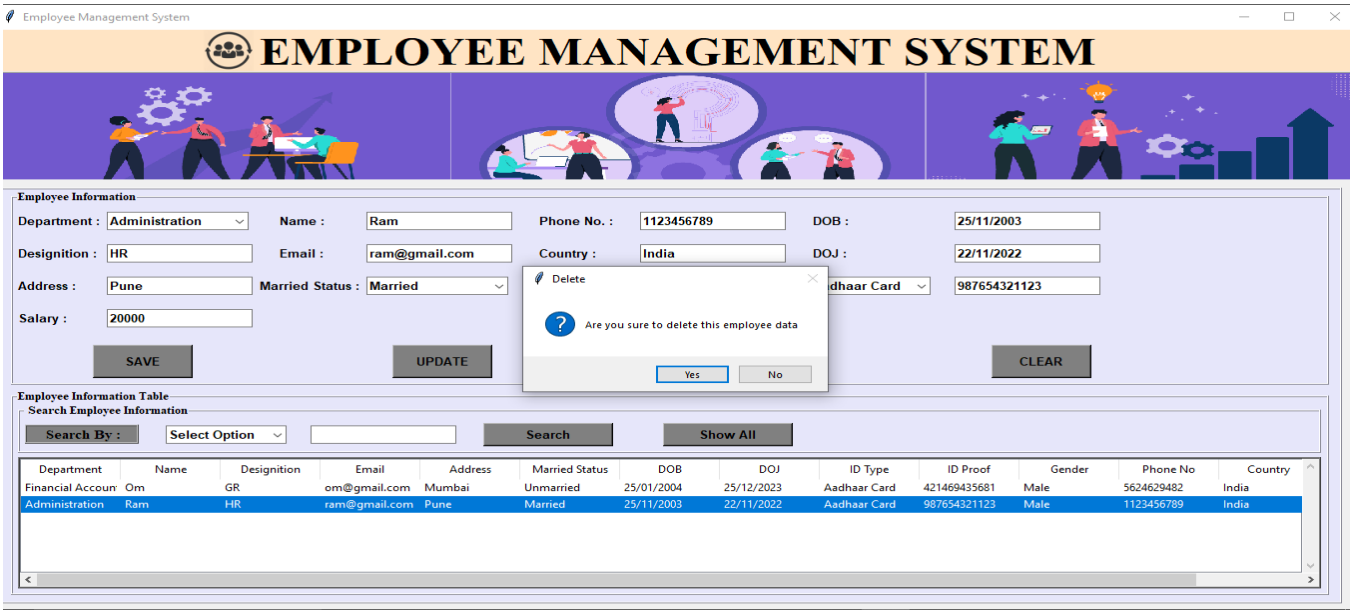


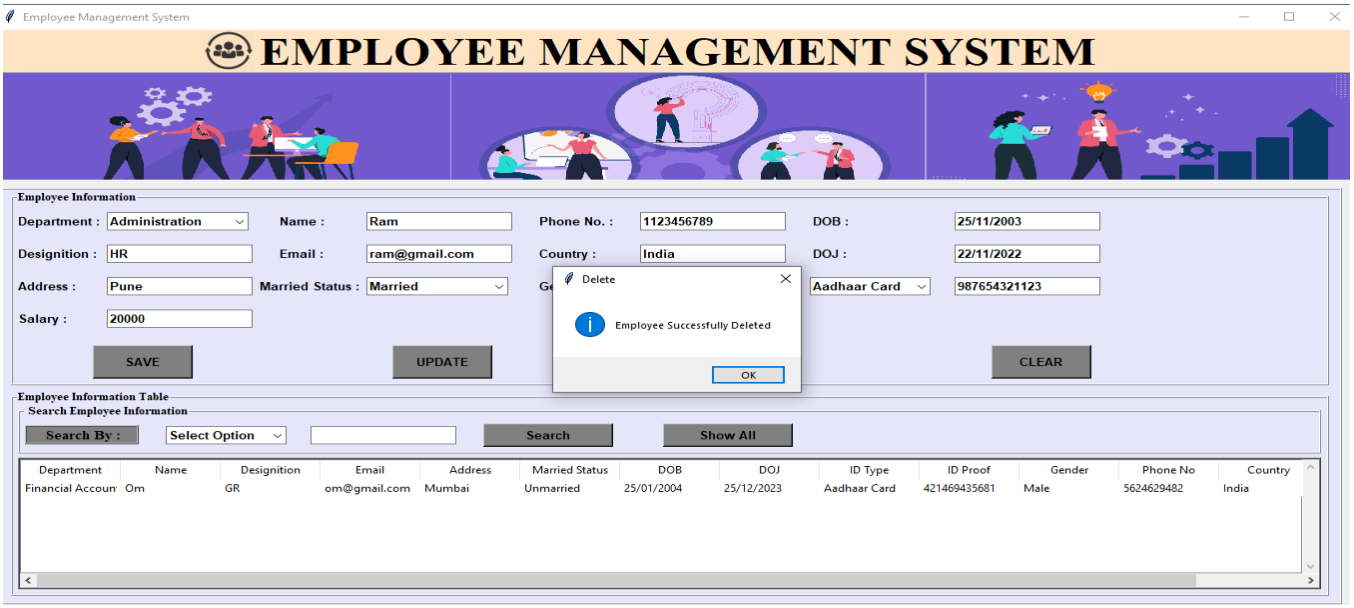
In this way, we can update records into the database.

3. Delete the Data From Database:

Mr. Om is currently selected as an employee which data we don't need anymore and want to release the database memory of it. The only thing has to be done is selecting the current person and pressing "Delete" button.



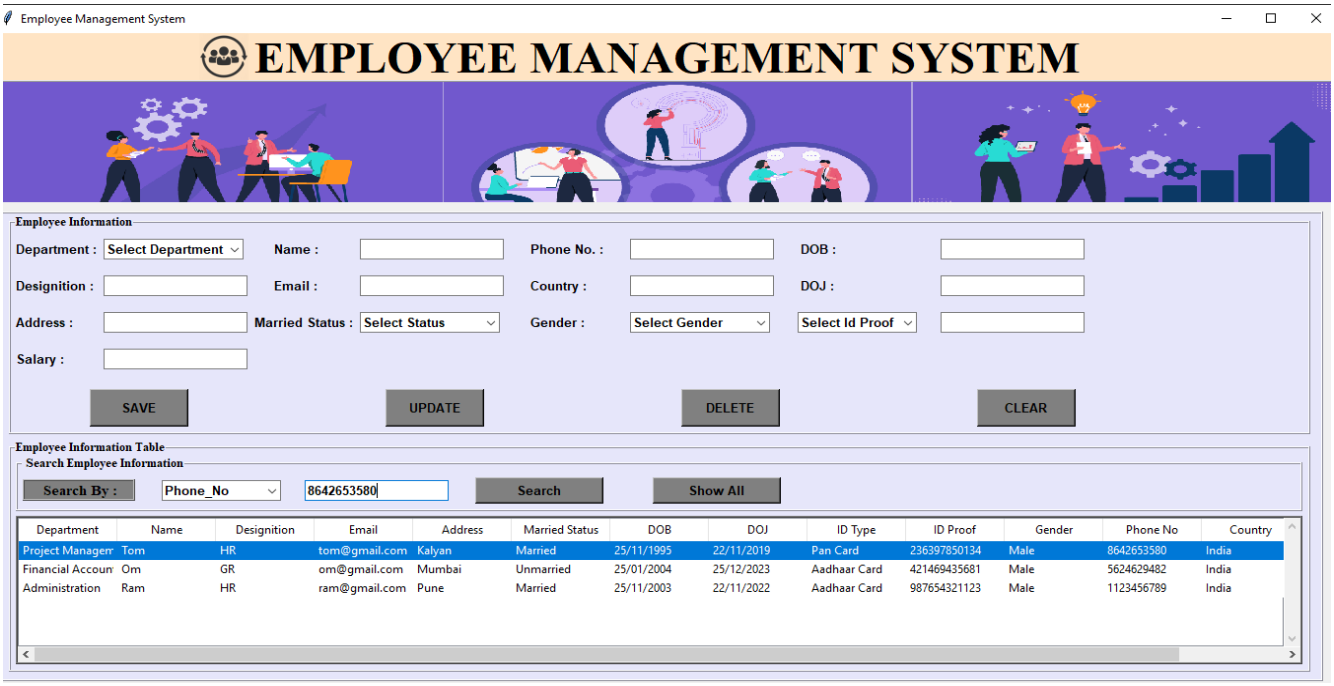




In this way, we can delete records from the database.

4. Search the data from Database:

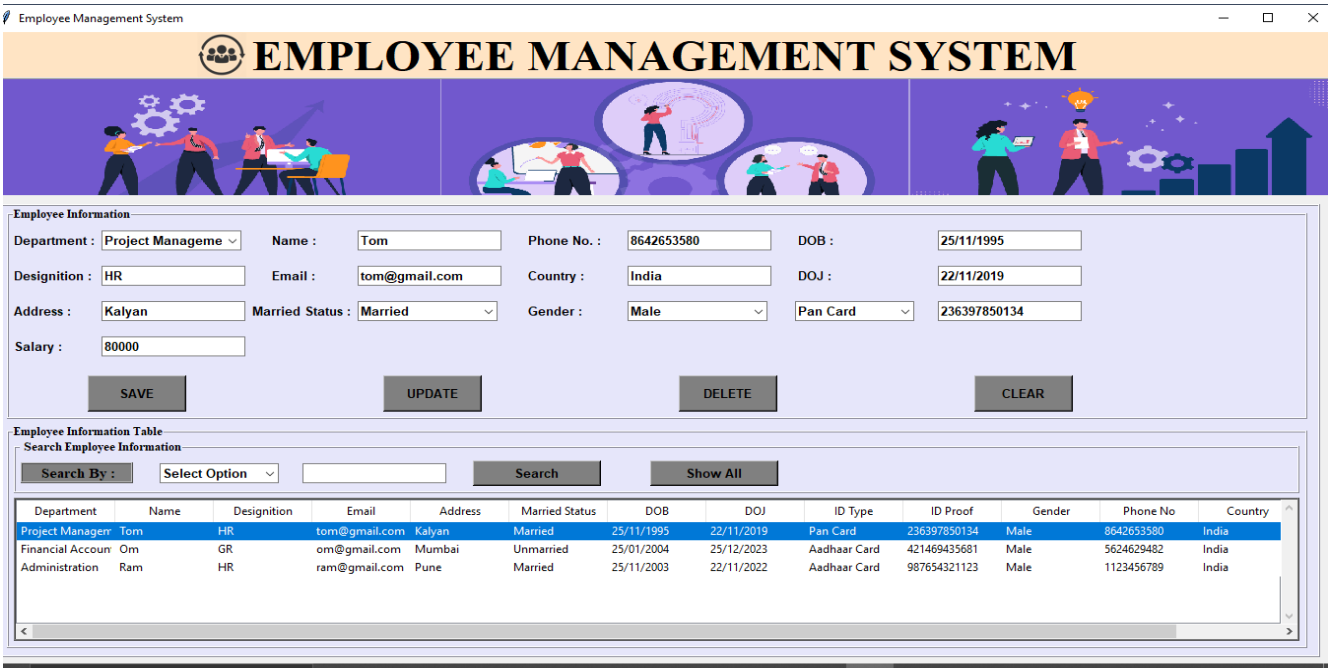
Here, we want to search the data of Mr. Tom by using his Phone_No . The only thing has to be done is enter his Phone_No and press the “Search” button.



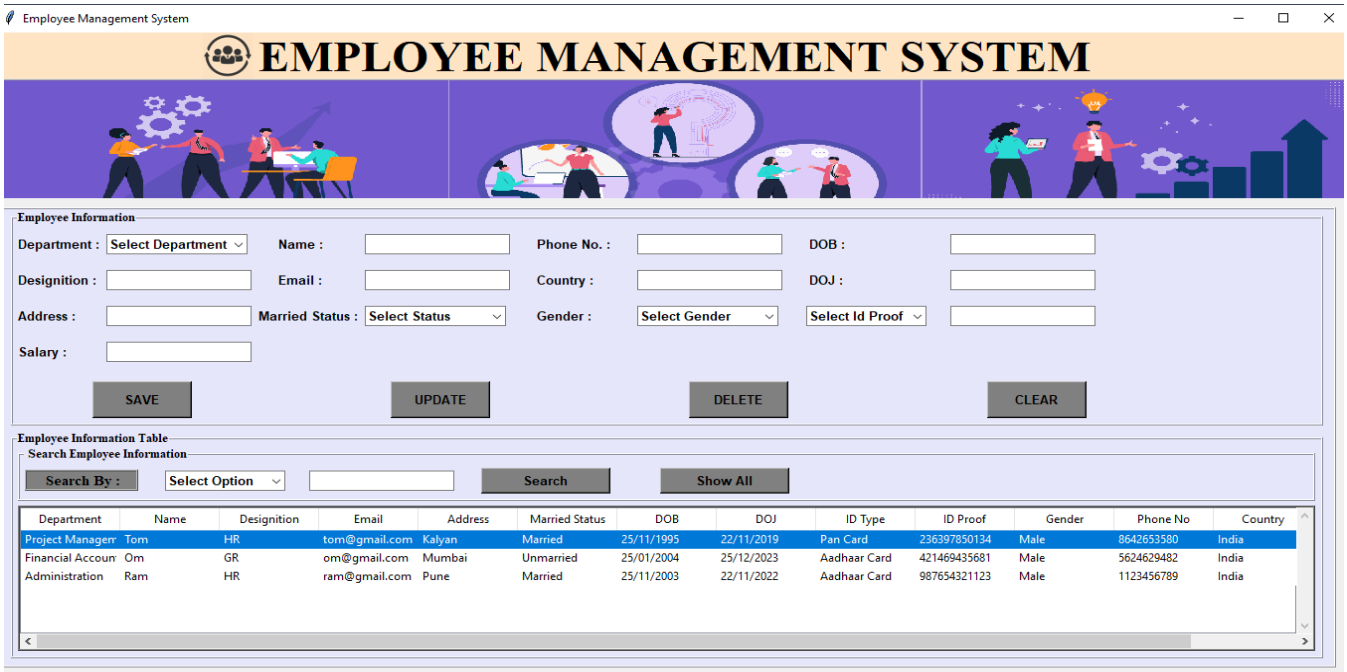
After Searching appropriate result is to be displayed.

In this way, we can search records from the database.

5. Clear the Data from Database:

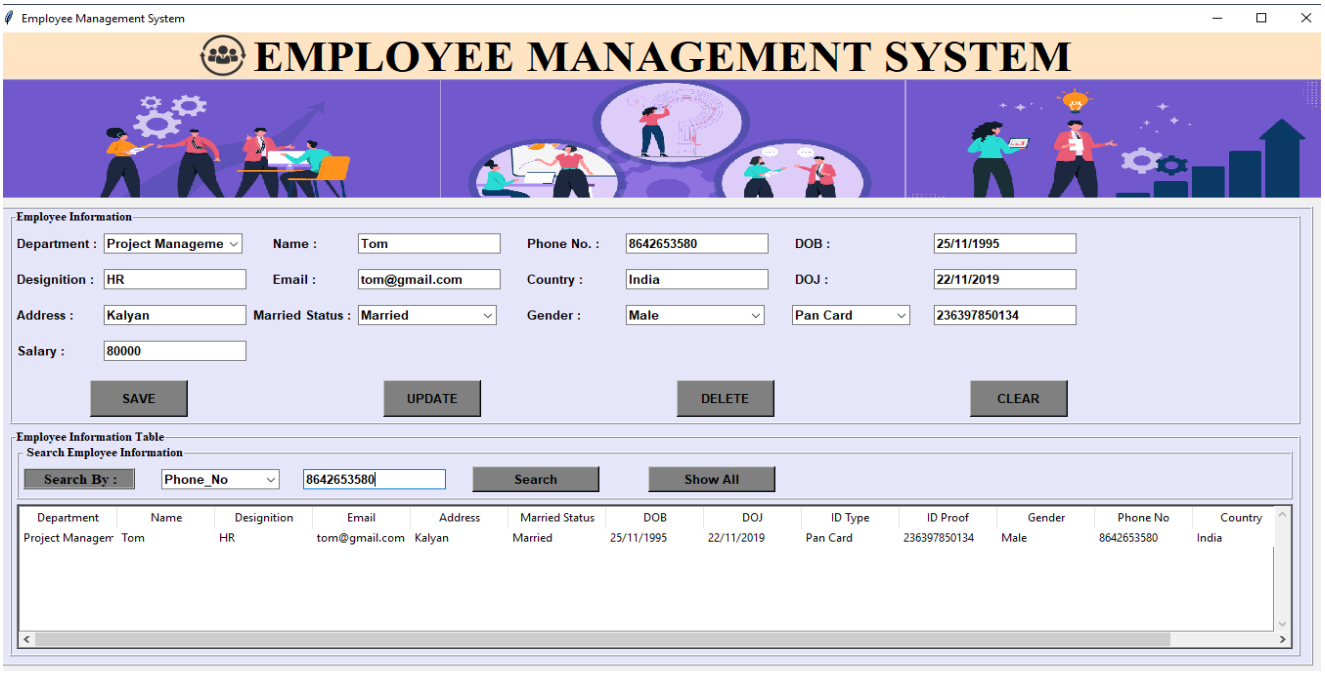


Here, we will able to clear all the data from Database by clicking “Clear” button.

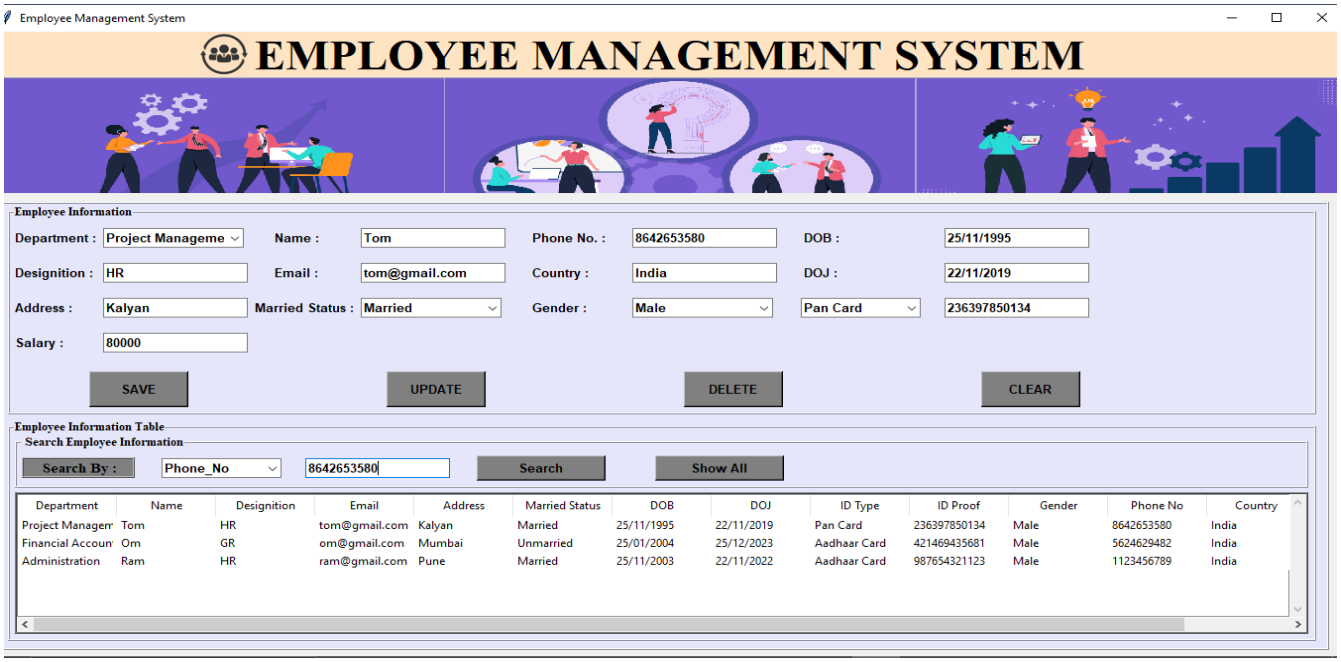


In this way, Data has been cleared.

6. Show All the Data from Database:



By clicking “Show All” button we will able to view all employee records.



In this way, we can view all records from the database.

CHAPTER NO 6

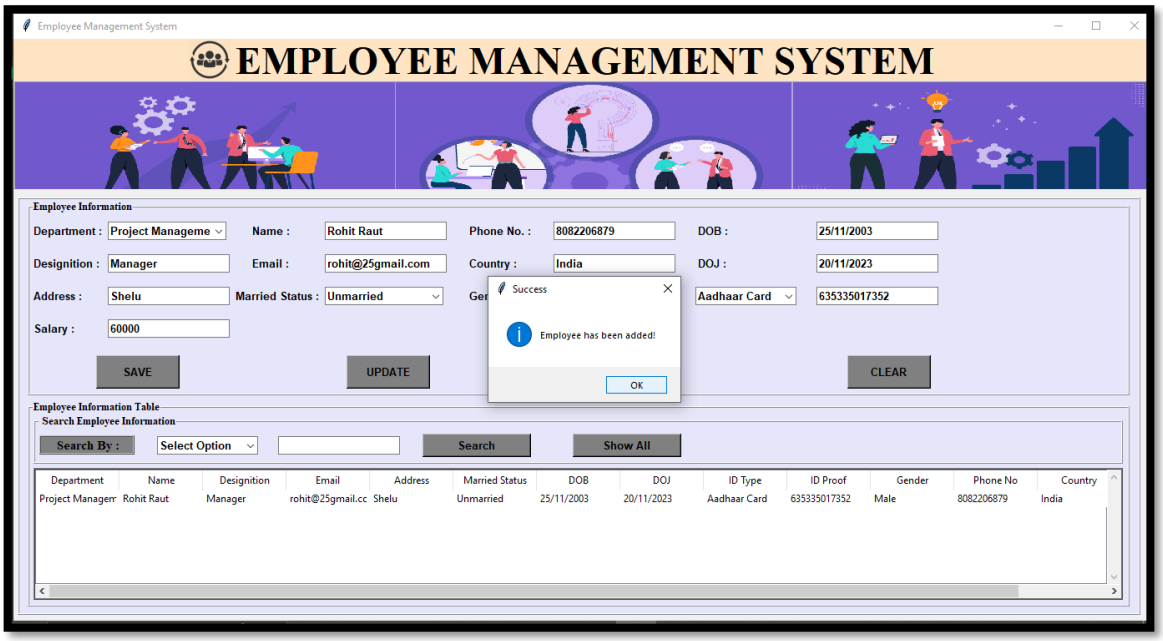
RESULT

RESULT

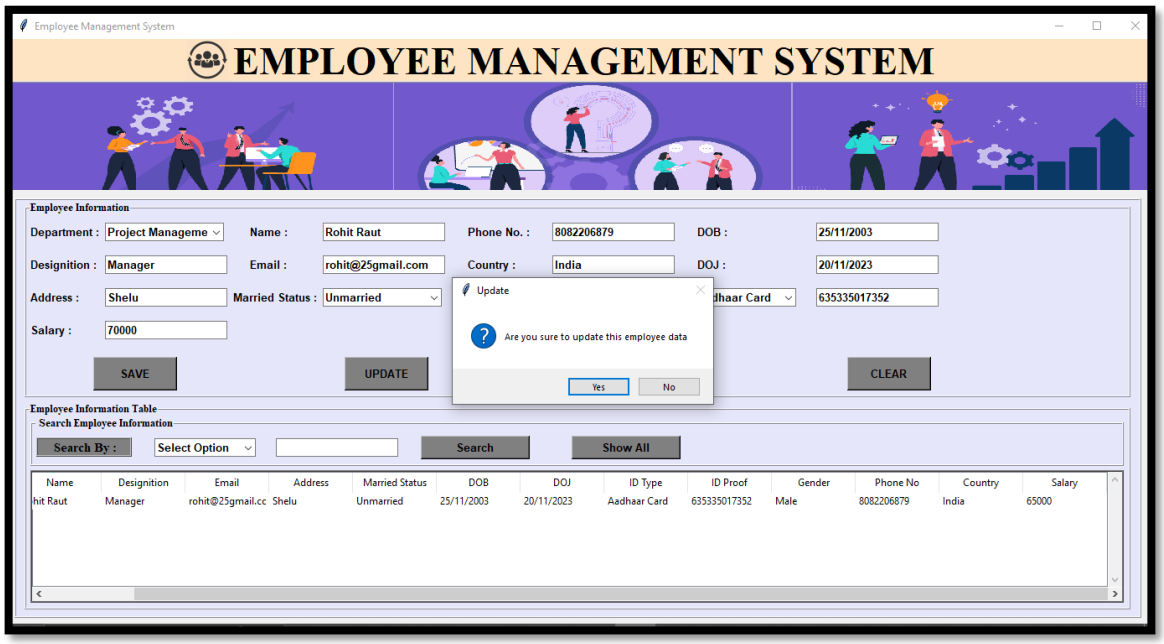
- **Graphical User Interface(GUI):**

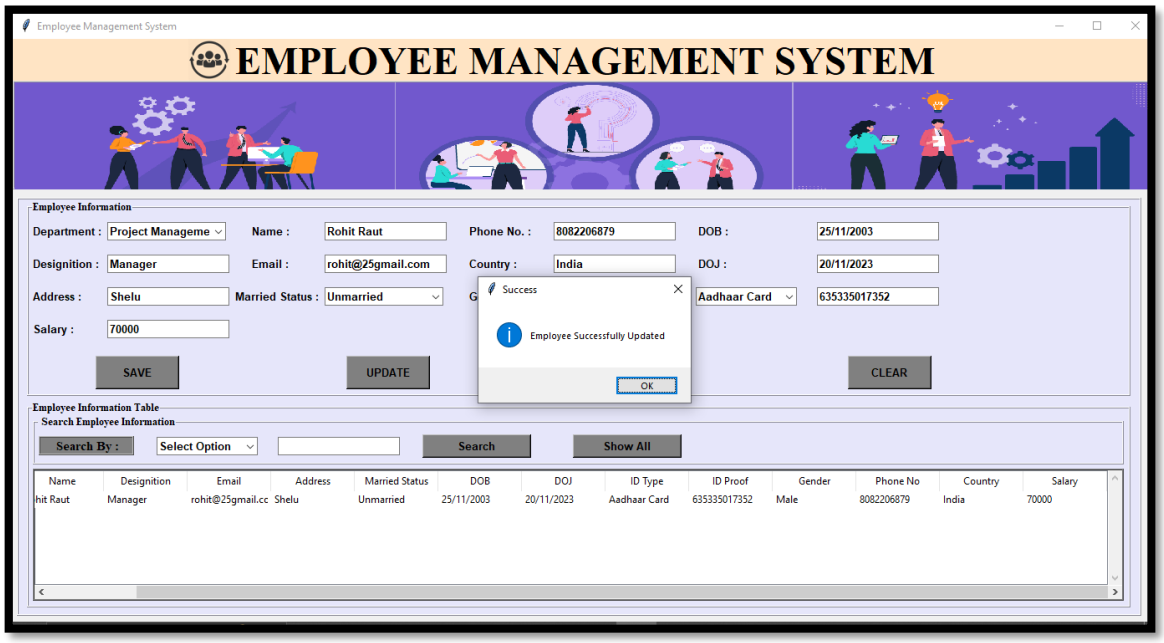
- **Save Data:**


Employee Management System
EMPLOYEE MANAGEMENT SYSTEM
Employee Information
Department : Project Manageme
Name : Rohit Raut
Phone No. : 8082206879
DOB : 25/11/2003
Designition : Manager
Email : rohit@25gmail.com
Country : India
DOJ : 20/11/2023
Address : Shelu
Married Status : Unmarried
Gender : Male
Aadhaar Card 635335017352
Salary : 60000
SAVE
UPDATE
DELETE
CLEAR
Employee Information Table
Search Employee Information
Search By :
Select Option
Search
Show All
Department Name Designition Email Address Married Status DOB DOJ ID Type ID Proof Gender Phone No Country

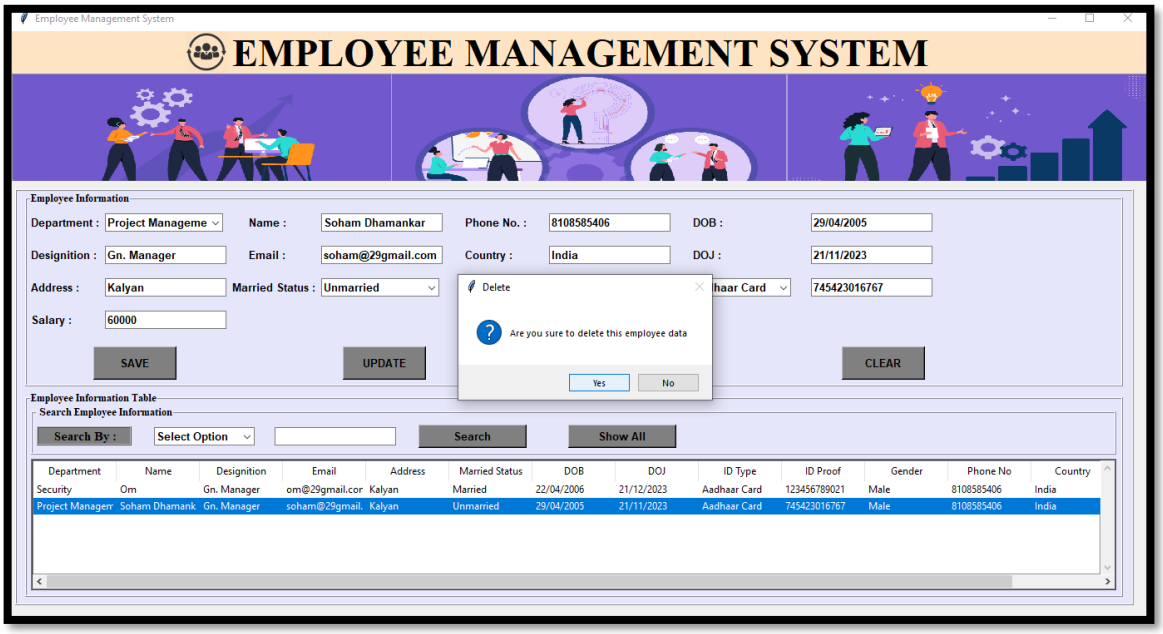

Employee Management System
EMPLOYEE MANAGEMENT SYSTEM
Employee Information
Department : Project Manageme
Name : Rohit Raut
Phone No. : 8082206879
DOB : 25/11/2003
Designition : Manager
Email : rohit@25gmail.com
Country : India
DOJ : 20/11/2023
Address : Shelu
Married Status : Unmarried
Aadhaar Card 635335017352
Salary : 60000
Success
Employee has been added!
OK
SAVE
UPDATE
CLEAR
Employee Information Table
Search Employee Information
Search By :
Select Option
Search
Show All
Department Name Designition Email Address Married Status DOB DOJ ID Type ID Proof Gender Phone No Country
Project Manager Rohit Raut Manager rohit@25gmail.cc Shelu Unmarried 25/11/2003 20/11/2023 Aadhaar Card 635335017352 Male 8082206879 India

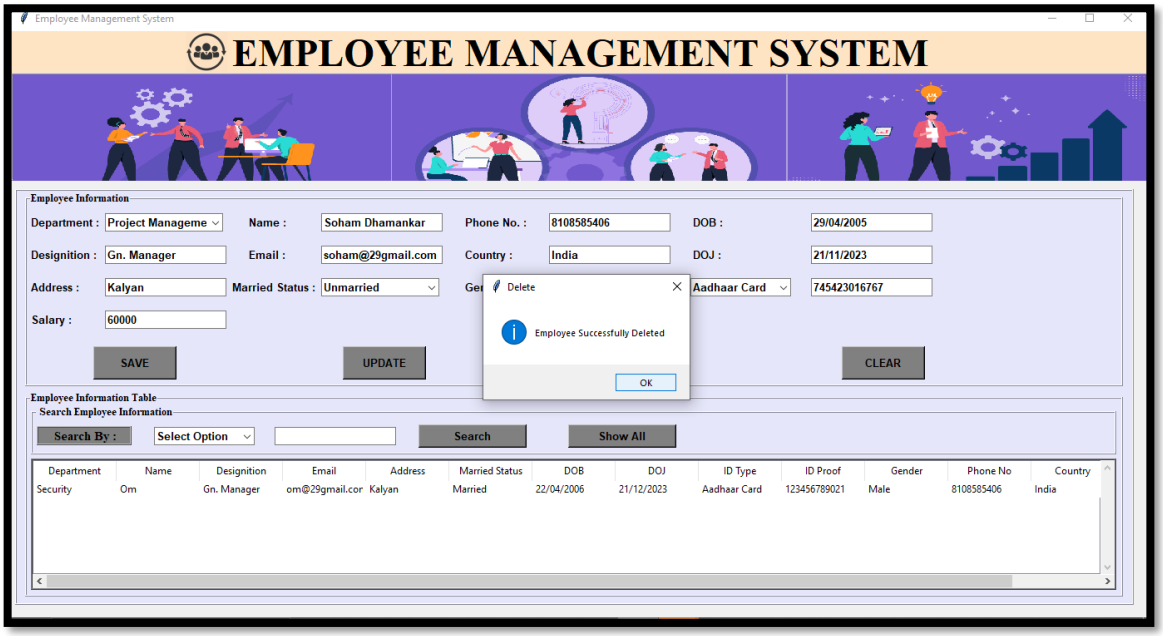
- **Update Data:**


Employee Management System
EMPLOYEE MANAGEMENT SYSTEM
Employee Information
Department : Project Manageme
Name : Rohit Raut
Phone No. : 8082206879
DOB : 25/11/2003
Designition : Manager
Email : rohit@25gmail.com
Country : India
DOJ : 20/11/2023
Address : Shelu
Married Status : Unmarried
635335017352
Salary : 70000
SAVE
UPDATE
CLEAR
Update
Are you sure to update this employee data
Yes
No
Employee Information Table
Search Employee Information
Search By :
Select Option
Search
Show All
Name Designition Email Address Married Status DOB DOJ ID Type ID Proof Gender Phone No Country Salary
hit Raut Manager rohit@25gmail.cc Shelu Unmarried 25/11/2003 20/11/2023 Aadhaar Card 635335017352 Male 8082206879 India 65000

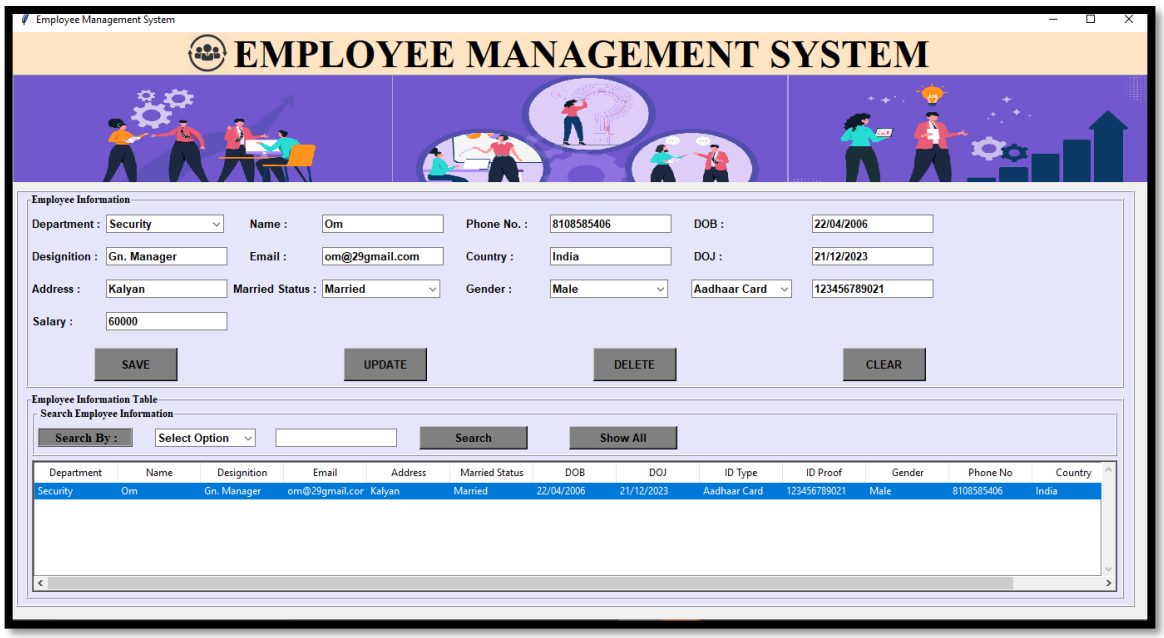

Employee Management System
EMPLOYEE MANAGEMENT SYSTEM
Employee Information
Department : Project Manageme
Name : Rohit Raut
Phone No. : 8082206879
DOB : 25/11/2003
Designition : Manager
Email : rohit@25gmail.com
Country : India
DOJ : 20/11/2023
Address : Shelu
Married Status : Unmarried
Aadhaar Card
635335017352
Salary : 70000
SAVE
UPDATE
CLEAR
Success
Employee Successfully Updated
OK
Employee Information Table
Search Employee Information
Search By :
Select Option
Search
Show All
Name Designition Email Address Married Status DOB DOJ ID Type ID Proof Gender Phone No Country Salary
hit Raut Manager rohit@25gmail.cc Shelu Unmarried 25/11/2003 20/11/2023 Aadhaar Card 635335017352 Male 8082206879 India 70000

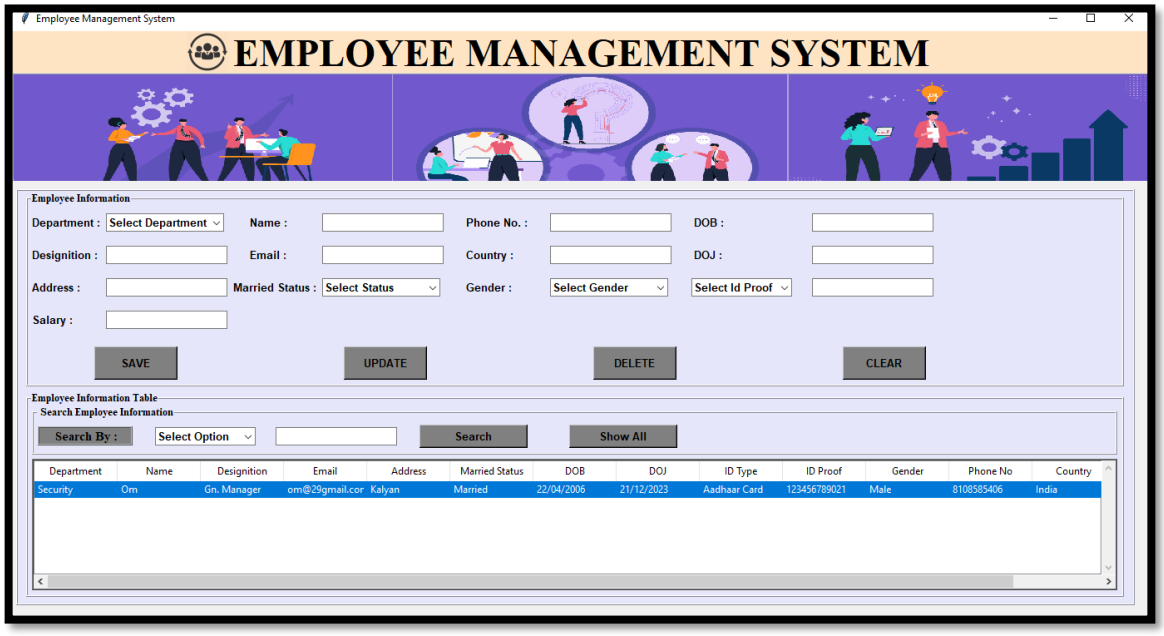
- **Delete Data:**


Employee Management System
EMPLOYEE MANAGEMENT SYSTEM
Employee Information
Department : Project Manageme
Name : Soham Dhamankar
Phone No. : 8108585406
DOB : 29/04/2005
Designition : Gn. Manager
Email : soham@29gmail.com
Country : India
DOJ : 21/11/2023
Address : Kalyan
Married Status : Unmarried
haar Card
745423016767
Salary : 60000
SAVE
UPDATE
CLEAR
Delete
Are you sure to delete this employee data
Yes
No
Employee Information Table
Search Employee Information
Search By :
Select Option
Search
Show All
Department Name Designition Email Address Married Status DOB DOJ ID Type ID Proof Gender Phone No Country
Security Om Gn. Manager om@29gmail.cor Kalyan Married 22/04/2006 21/12/2023 Aadhaar Card 123456789021 Male 8108585406 India
Project Manager Soham Dhamank Gn. Manager soham@29gmail. Kalyan Unmarried 29/04/2005 21/11/2023 Aadhaar Card 745423016767 Male 8108585406 India

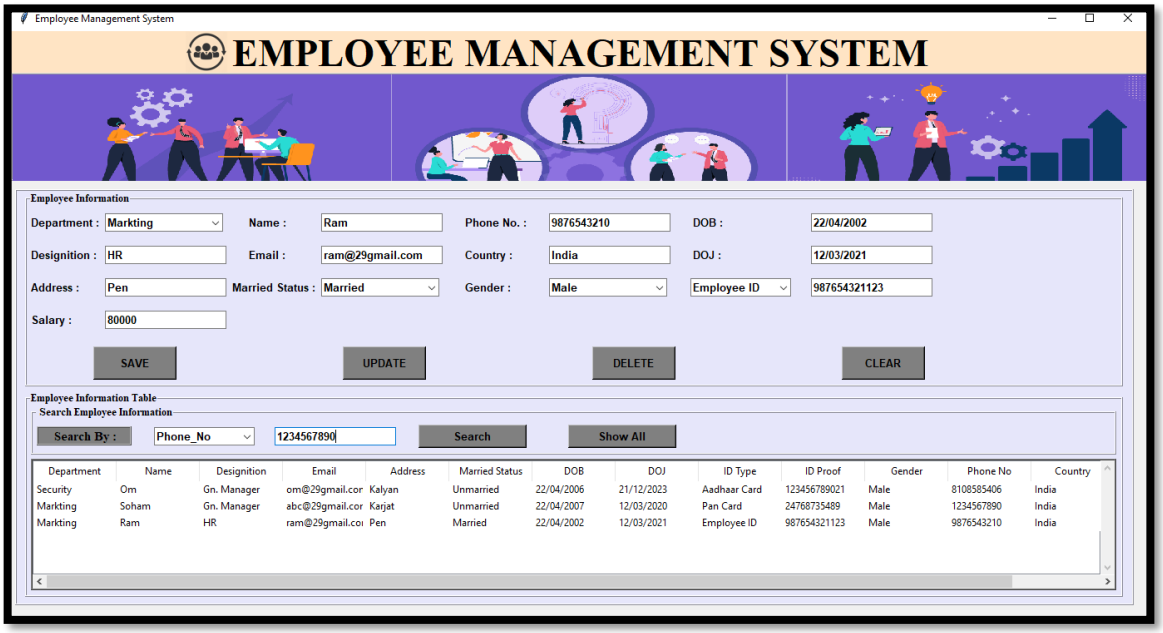

Employee Management System
EMPLOYEE MANAGEMENT SYSTEM
Employee Information
Department : Project Manageme
Name : Soham Dhamankar
Phone No. : 8108585406
DOB : 29/04/2005
Designition : Gn. Manager
Email : soham@29gmail.com
Country : India
DOJ : 21/11/2023
Address : Kalyan
Married Status : Unmarried
Aadhaar Card
745423016767
Salary : 60000
SAVE
UPDATE
CLEAR
Delete
Employee Successfully Deleted
OK
Employee Information Table
Search Employee Information
Search By :
Select Option
Search
Show All
Department Name Designition Email Address Married Status DOB DOJ ID Type ID Proof Gender Phone No Country
Security Om Gn. Manager om@29gmail.cor Kalyan Married 22/04/2006 21/12/2023 Aadhaar Card 123456789021 Male 8108585406 India

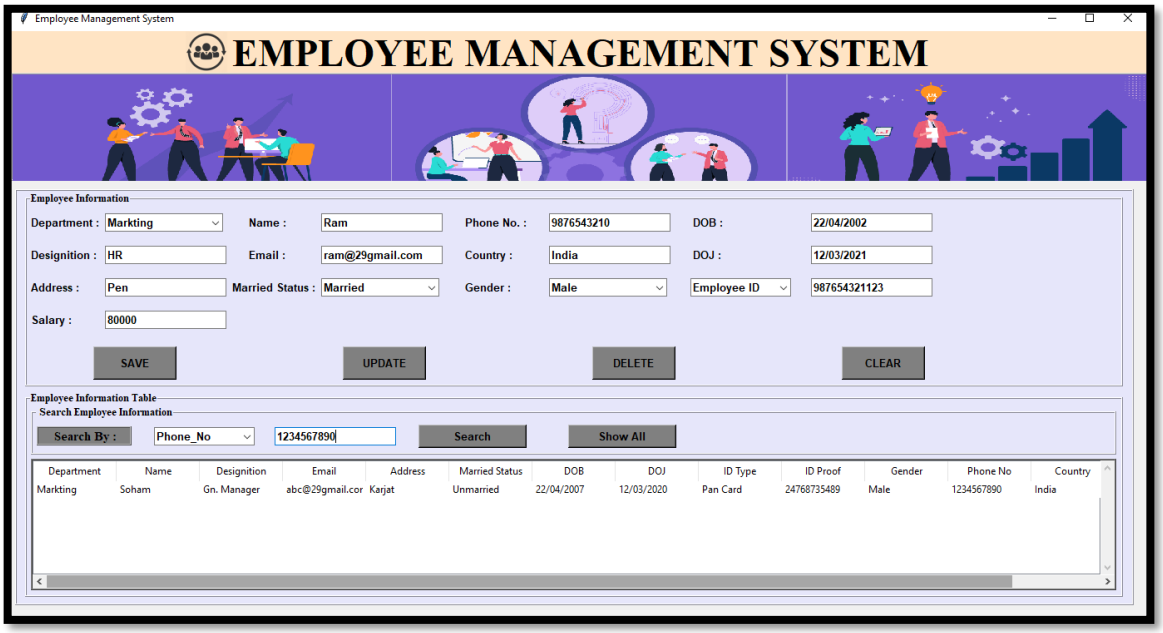
- **Clear Data:**


Employee Management System
EMPLOYEE MANAGEMENT SYSTEM
Employee Information
Department : Security
Name : Om
Phone No. : 8108585406
DOB : 22/04/2006
Designition : Gn. Manager
Email : om@29gmail.com
Country : India
DOJ : 21/12/2023
Address : Kalyan
Married Status : Married
Gender : Male
Aadhaar Card
123456789021
Salary : 60000
SAVE
UPDATE
DELETE
CLEAR
Employee Information Table
Search Employee Information
Search By :
Select Option
Search
Show All
Department Name Designition Email Address Married Status DOB DOJ ID Type ID Proof Gender Phone No Country
Security Om Gn. Manager om@29gmail.cor Kalyan Married 22/04/2006 21/12/2023 Aadhaar Card 123456789021 Male 8108585406 India


Employee Management System
EMPLOYEE MANAGEMENT SYSTEM
Employee Information
Department : Select Department
Name :
Phone No. :
DOB :
Designition :
Email :
Country :
DOJ :
Address :
Married Status : Select Status
Gender : Select Gender
Select Id Proof
Salary :
SAVE
UPDATE
DELETE
CLEAR
Employee Information Table
Search Employee Information
Search By :
Select Option
Search
Show All
Department Name Designition Email Address Married Status DOB DOJ ID Type ID Proof Gender Phone No Country
Security Om Gn. Manager om@29gmail.cor Kalyan Married 22/04/2006 21/12/2023 Aadhaar Card 123456789021 Male 8108585406 India

- **Search:**


Employee Management System
EMPLOYEE MANAGEMENT SYSTEM
Employee Information
Department : Markting
Name : Ram
Phone No. : 9876543210
DOB : 22/04/2002
Designition : HR
Email : ram@29gmail.com
Country : India
DOJ : 12/03/2021
Address : Pen
Married Status : Married
Gender : Male
Employee ID 987654321123
Salary : 80000
SAVE
UPDATE
DELETE
CLEAR
Employee Information Table
Search Employee Information
Search By :
Phone_No
1234567890
Search
Show All
Department Name Designition Email Address Married Status DOB DOJ ID Type ID Proof Gender Phone No Country
Security Om Gn. Manager om@29gmail.cor Kalyan Unmarried 22/04/2006 21/12/2023 Aadhaar Card 123456789021 Male 8108585406 India
Markting Soham Gn. Manager abc@29gmail.cor Karjat Unmarried 22/04/2007 12/03/2020 Pan Card 24768735489 Male 1234567890 India
Markting Ram HR ram@29gmail.cor Pen Married 22/04/2002 12/03/2021 Employee ID 987654321123 Male 9876543210 India


Employee Management System
EMPLOYEE MANAGEMENT SYSTEM
Employee Information
Department : Markting
Name : Ram
Phone No. : 9876543210
DOB : 22/04/2002
Designition : HR
Email : ram@29gmail.com
Country : India
DOJ : 12/03/2021
Address : Pen
Married Status : Married
Gender : Male
Employee ID 987654321123
Salary : 80000
SAVE
UPDATE
DELETE
CLEAR
Employee Information Table
Search Employee Information
Search By :
Phone_No
1234567890
Search
Show All
Department Name Designition Email Address Married Status DOB DOJ ID Type ID Proof Gender Phone No Country
Markting Soham Gn. Manager abc@29gmail.cor Karjat Unmarried 22/04/2007 12/03/2020 Pan Card 24768735489 Male 1234567890 India

CHAPTER NO.7

FUTURE ENHANCEMENTS

FUTURE ENHANCEMENTS

As a future work, some additional stuff could be implemented and integrated into the application code making it much more reliable and flexible; especially what concerns a pay-roll module, for instance.

Apparently, the role of such systems is basic and essential within each company that wants to keep a really good control and record concerning its personnel data, functionality and performance on all levels in its structure. Every organization, in nowadays, has the necessity of managing its staff on a really good level as the staff has definitely the greatest merit of building up a company as such as it is.

The well managed employee means giving the appropriate financial award-ness and all kind of benefits as such as they have been deserved. That's why the development of such systems is not just a programming business a lot of people are ordinarily involved in such projects and one of the basic requirements is the reliability of the system, especially what concerns the storage of data and all of the operations that will be performed upon it.

CHAPTER NO.8

CONCLUSION

CONCLUSION

The Employee Management System (EMS) project developed using Python with MySQL database has successfully achieved its aim and objectives. This project provides basic essential functions, within each company that wants to keep a really good control and record concerning its personal data, functionality and performance on all levels in its structure.

Throughout the project, best coding practices were followed, ensuring code efficiency, readability, and maintainability. Extensive testing and validation were conducted to ensure the system webpage's reliability and accuracy.

This Employee Management system in python serves as a valuable for companies and organization. It demonstrates the power of python programming in creating functional and efficient software for data management applications.

CHAPTER NO.9

REFERENCE

REFERENCE

- Core Python Programming by Dr. Nageswara Rao
- Think Python
- Python GUI with MySQL: A step by step Guide to Database Programming
- Implementation of MySQL in Python
- Python Programming 2E by Reema Thareja