# INTERDISCIPLINARY PROJECT REPORT

**at**

# Sathyabama Institute of Science and Technology (Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering

By

MAKINEEDI SOWMYA SRI LAKSHMI BHAVANI

**(REG.NO: 40110698)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SCHOOL OF COMPUTING**

**SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY**

### (DEEMED TO BE UNIVERSITY)

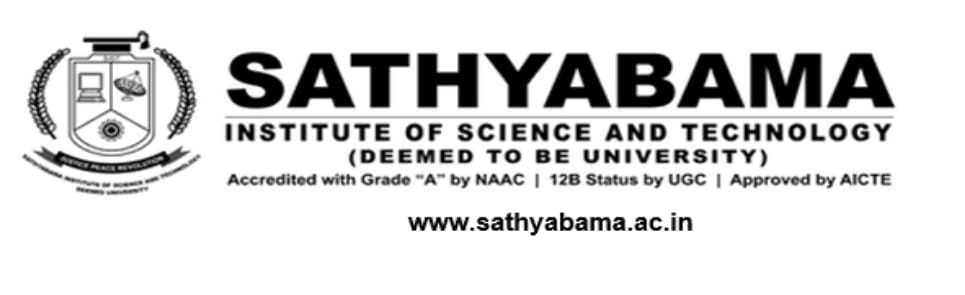
**Accredited with Grade “A” by NAAC | 12 B Status**

**by UGC | Approved by AICTE**

**JEPPIAR NAGAR, RAJIV GANDHISALAI,**

**CHENNAI – 600119**

**APRIL 2023**



# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of **MAKINEEDI SOWMYA SRI LAKSHMI BHAVANI (40110698)** who carried out the project entitled “**EMPLOYEE MANAGEMENT SYSTEM**” under my supervision from FEB 2023 to APRIL 2023.

## Internal Guide

## Mrs. B. Sandhiya M.E, (Ph.D.)

**Head of the Department**

**Dr . L. Lakshmanan M.E., Ph.D.**



## Submitted for Viva voce Examination held on

**Internal Examiner External Examiner**

**DECLARATION**

I, **MAKINEEDI SOWMYA SRI LAKSHMI BHAVANI (REG. NO: 40110698)** hereby declare that the Project Report entitled **“EMPLOYEE MANAGEMENT SYSTEM”** done by me under the guidance of **Mrs. B. Sandhiya M.E, (Ph.D.)** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

**DATE:**

**PLACE: SIGNATURE OF THE CANDIDATE**

**ACKNOWLEDGEMENT**

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D.**, **Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D., and Dr.S.Vigneshwari M.E., Ph.D., Heads of the Department** of **Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mrs. B. Sandhiya M.E, (Ph.D.)** her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

# ABSTRACT

This **Employee Management System** created based on python, Tkinter, and MYSQL Database. The Customer Feedback Management System is a small project for providing employee details of a company. Every organisation whether government or private uses an information system to store data of their staff. However, in India it is found that many small scale industries use pen and paper to keep a record.

This system will calculate the salary of them at the end of month. It also calculates overtime and total working hours of each employee. It saves lots of time and has no error in pay calculation hence preventing clashes between HR Team and employees. So that both employer and employee can focus on their work to develop their company.

A Employee Management System in Python is a simple management system that includes a employee details, such as first name, last name, gender, age and dob .This project mainly focus on **Save**, **clear**, **Update** and **Delete**. There’s an external database connection file used in this mini project to save employee data permanently.

This Employee Management System Project in Python is a basic Python project that aims to help businesses manage their loyal and valuable employee.

|  |  |  |
| --- | --- | --- |
|  | **TABLE OF CONTENTS** |  |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | 1 |
|  | **LIST OF FIGURES** | **3** |
| **1.** | **INTRODUCTION** | **4** |
|  | 1.1 Graphical User Interface | 5 |
|  | 1.2 Database | 6 |
| **2.** | **AIM AND SCOPE OF THE PRESENT INVESTIGATION** | **8** |
|  | 2.1 Aim of the employee management system | 8 |
|  | 2.2 Scope of the employee management system | 9 |

|  |  |  |
| --- | --- | --- |
| **3.** | **EXPRIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED**   * 1. Importing required Libraries   2. creating user interface   3. data database connectivity | **10**  10  11  11 |
| **4.** | **SUMMARY AND CONCLUSIONS** | **14** |
| **5.** | **REFERENCES**  **SOURCE CODE EXPLANATION SOURCE CODE**  **SCREEN SHOTS** | **15**  16  23  34 |

# LIST OF FIGURES

**FIGURENO FIGURENAME PAGENO**

|  |  |  |
| --- | --- | --- |
| 1.1.1 | Tkinter (GUI) | 5 |
| 1.2.1 | Database | 6 |
| 1.2.2 | MySQL | 7 |
| 2.1.1  2.1.2 | EMS  Architecture | 8  9 |
| 3.3.1 | Database Connectivity | 13 |
| 5.4.1 | Employee Data Base | 14 |
| 5.4.2 | Adding New Employee | 15 |
| 5.4.3 | Updating Details | 15 |
| 5.4.4 | Deleting Details | 16 |

## CHAPTER - 1

**INTRODUCTION**

This simple Employee Management system project is written in Python. The project file contains a python script. This is a simple GUI based application which is very easy to understand and use. It uses Tkinter module for the GUI. Talking about the application, the user just has to enter his/her name, gender, age and salary marital status regarding employee. This is a simple GUI application, that does not create an external file or uses external files as a database to store the data permanently.

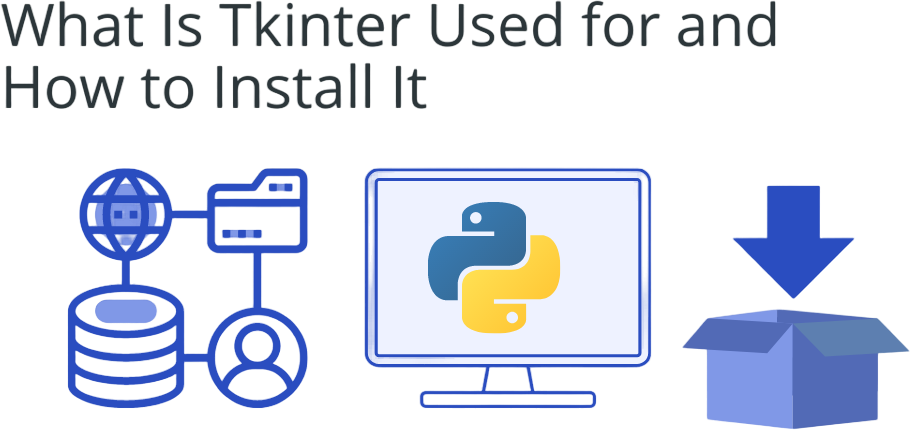
## Graphical user interface (GUI):

A graphical user interface is an application that has buttons, windows, and lots of other widgets that the user can use to interact with your application. A good example would be a web browser. It has buttons, tabs, and a main window where all the content loads.

Tkinter is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact, it's the only framework built into the Python standard library

It refers to a type of interface that allows users to interact with digital devices or software applications through graphical elements such as icons, buttons, menus, and windows, rather than using text commands.

A GUI makes it easier for users to perform tasks on a computer or device by providing a visual representation of data and actions. It allows users to navigate, manipulate and interact with information and applications through intuitive and easy-to-use visual elements.



## Fig 1.1.1 Tkinter (GUI)

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

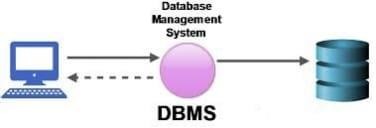
This framework provides Python users with a simple way to create GUI elements using the widgets found in the Tk toolkit. Tk widgets can be used to construct buttons, menus, data fields, etc. in a Python application. Once created, these graphical elements can be associated with or interact with features, functionality, methods, data or even other widgets.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

* Import the Tkinter module.
* Create the GUI application main window.
* Add one or more of the above-mentioned widgets to the GUI application.

* Enter the main event loop to take action against each event triggered by the user

## Database:



**Fig 1.2.1 DataBase**

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.

Data within the most common types of databases in operation today is typically modeled in rows and columns in a series of tables to make processing and data querying efficient. The data can then be easily accessed, managed, modified, updated, controlled, and organized. Most databases use structured query language (SQL) for writing and querying data

There are many databases available like MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server, etc

## MYSQL:

**Fig 1.2.3 MYSQL**

MYSQL A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and “pointers” between different tables. The database enforces these rules, so that with a well-designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data.

## CHAPTER 2

**AIM AND SCOPE OF THE PRESENT INVESTIGATION**

## Aim of the employee management system:



**Fig 2.1. EMS**

This GUI based Employee Management system provides the simplest management of employee details. In short, this projects mainly focus on save, delete, clear, update. There’s an external database connection file used in this mini project to save empoyee’s data permanently.

This Employee Management System Project in Python is a basic Python project that aims to help businesses manage their loyal and valuable employees.

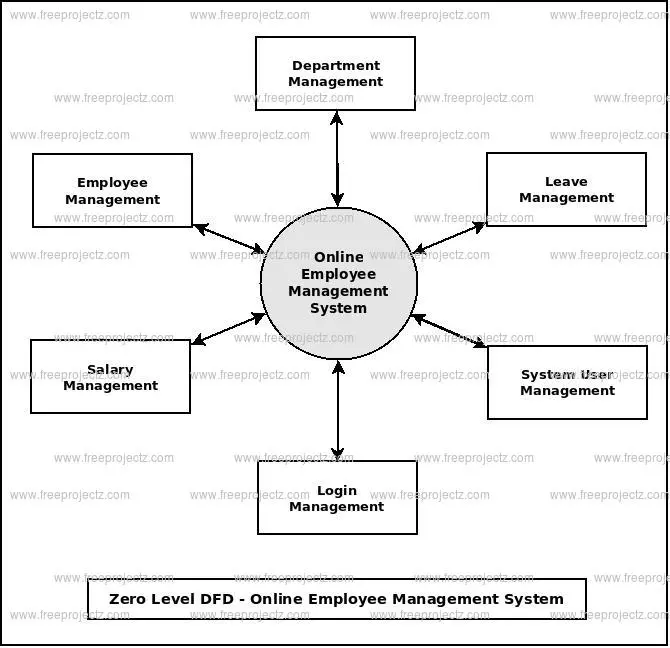


Fig 2.2. Architecture

## Scope of the system to manage business:

An effective employee management system is used to manage the work process and organizational responsibilities of human resources and other departments. It helps managers and employees to work together and accurately monitor, access, manage, and efficiently utilize the working hours for better business growth .

* + - To develop a system that will serve as the centralized database of customer feedback.
    - Companies will be able to use the system as a source of data .
    - The system’s output will allow businesses to improve the quality of their services

## CHAPTER 3

**EXPRIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED**

## Importing required Libraries:

****

**tkinter:** Python Tkinter is the most preferred package used for creating nice GUIs for applications as it has a variety of methods like pack(), grid(), and place() for geometry management. It has standard attributed dimensions, fonts, colors, cursors, anchors, and bitmaps for better GUI. Moreover, it has a vast array of widgets to choose from and is by far the easiest to use. The combination of all these features makes Python Tkinter makes it very popular among Python developers and makes it a favorable tool to use.

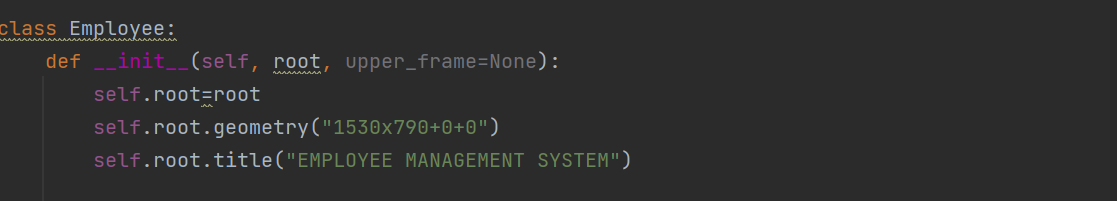
**Ttk and tkMessageBox:** The ttk and tkMessageBox module is used to display message boxes in your applications. This module provides a number of functions that you can use to display an appropriate message.

**Pillow (a fork of PIL):**

Python Imaging Library (expansion of PIL) is the de facto image processing package for Python language. It incorporates lightweight image processing tools that aids in editing, creating and saving images. Support for Python Imaging Library got discontinued in 2011, but a project named pillow forked the original PIL project and added Python3.x support to it. Pillow was announced as a replacement for PIL for future usage. Pillow supports a large number of image file formats including BMP, PNG, JPEG, and TIFF. The library encourages adding support for newer formats in the library by creating new file decoders.

This module is not preloaded with Python. So to install it execute the following command in the command-line: **pip install pillow**

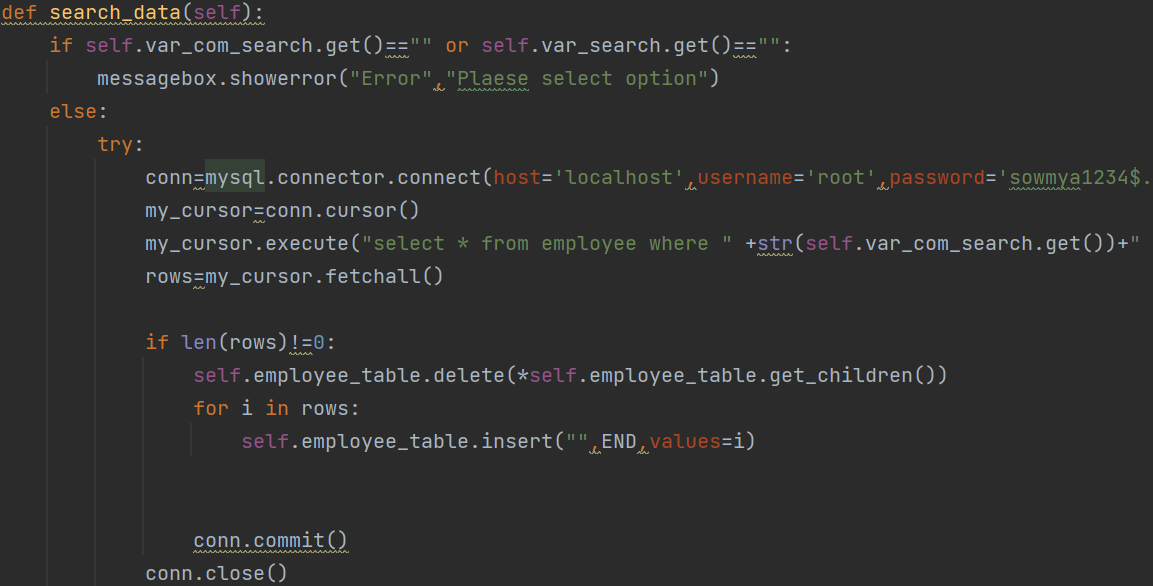
## Creating user interface:

****

Import the Tkinter module. Create the GUI application main window. Add one or more of the above-mentioned widgets to the GUI application. Enter the main event loop to take action against each event triggered by the user.

Creating the Employee details interface. For adding the employee details entering interface and updating details interface finally it saves database these all created

## Data database connectivity:

****

1. **Import MySQL connector module**

import MySQL connector statement to connect with MySQL module in the program. Using the classes and methods defined in the MySQL module we can communicate with the MySQL database.

## Use the connect() method

Use the connect() method of the connector class with the database name. To establish a connection to MySQL, you need to pass the database name you want to connect. If you specify the database file name that already presents on the disk, it will connect to it. But if your specified MySQL database file doesn’t exist, MySQL creates a new database for you.

This method returns the MySQL Connection Object if the connection is successful.

## Use the cursor() method

Use the cursor() method of a connection class to create a cursor object to execute MYSQL command queries from Python.

## Use the execute() method

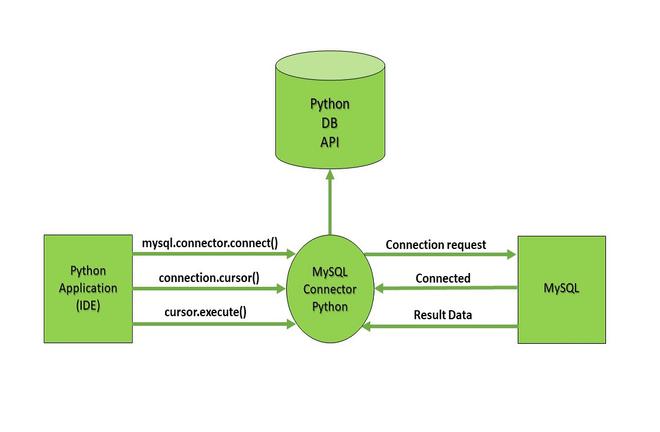
The execute() methods run the SQL query and return the result.

## Extract result using fetchall()

Use cursor.fetchall() or fetchone() or fetchmany() to read query result.

## Close cursor and connection objects

use cursor.clsoe() and connection.clsoe() method to close the cursor and MySQL connections after your work completes



## Fig 3.3.1 Database Connectivity

Connect to MySQL Database in Python

Import MySQl connector package. Create a connection object using the connect() method by passing the name of the database as a parameter to it. The cursor() method returns a cursor object using which you can communicate with MySQL

First, connect to the MySQL database by creating a Connection object. Second, create a Cursor object by calling the cursor method of the Connection object. Third, execute an INSERT statement.

## CHAPTER 4 SUMMARY AND CONCLUSIONS

**SUMMARY:**

The Employee Management System Project In Python was developed using Python Programming, this project created using Graphical User Interface (GUI) and connected into database using MySQL

This EMS can help you improve team motivation and productivity.The tool lets you monitor, evaluate and provide workers with feedback on their performance. When employee receive timely feedback, it helps in improving the work culture.

## CONCLUSIONS:

The Employee Management System Project In Python is a graphical user interface system written and designed in the Python programming language.This is a simple design which allows you to insert all the necessary data like Name, dob, gender, age, salary Department and Destignition.

## CHAPTER 5 REFERENCES AND SCREEN SHOTS

**REFERENCES:**

## Adam L. Darlow & Steven A. Sloman. 2010. “Two systems of reasoning: Architecture and relation to emotion,” WIREs Cognitive Science, 1: 382-392.

## Christopher L. Aberson, Michael Healy, & Victoria Romero. 2000. Ingroup Bias and Self-Esteem: A Meta-Analysis. Personality and Social Psychology Review, 4: 157-173.

## Eccles, R. G. & Nohria, N. (1992). Beyond the Hype: Rediscovering the Essence of Management. Boston: The Harvard Business School Press, p. 47.

## Elizabeth Kolbert. 2017. Why Facts Don’t Change our Minds. The New Yorker, February 27, 2017.

## Hannaway, J. (1989). Managers Managing: The Workings of an Administrative System. New York: Oxford University Press, P. 39.

## Jennifer M. George. 2000. “Emotions and leadership: The role of emotional intelligence.” Human Relations, 53, 1027-1055.

## Kotter, J. P. (1999). “What Effective General Managers Really Do,” Harvard Business Review, March–April 1999, pp. 145–159.

## Linda K. Trevino & Michael E. Brown. 2004. Managing to be ethical: Debunking five business ethics myths. Academy of Management Executive, 18: 69-81.

## Source Code Explanation:

from tkinter import\*  
from tkinter import ttk  
from PIL import Image,ImageTk  
import mysql.connector  
from tkinter import messagebox  
from tkinter import filedialog  
from time import strftime  
from datetime import datetime  
import os  
  
class Employee:  
 def \_\_init\_\_(self, root, upper\_frame=None):  
 self.root=root  
 self.root.geometry("1530x790+0+0")  
 self.root.title("EMPLOYEE MANAGEMENT SYSTEM")

import required libraries and the create Message box using Tkinter module and set height and width according to the screen height and width

# Variables  
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\_idproofcomb=StringVar()  
self.var\_idproof=StringVar()  
self.var\_gender=StringVar()  
self.var\_phone=StringVar()  
self.var\_country=StringVar()  
self.var\_salary=StringVar()

Create variables for future use

def get\_cursor(self,event=""):  
 cursor\_row=self.employee\_table.focus()  
 content=self.employee\_table.item(cursor\_row)  
 data=content["values"]

Create a function database create a database using MySQL and connect with the database and create column rows.

def add\_data(self):  
 if (self.var\_dep.get()=="" or self.var\_email.get()=="" or self.var\_idproof.get()==""):  
 messagebox.showerror("Error","All Fields Are required")  
 else:  
 try:  
 conn=mysql.connector.connect(host="localhost",username="root",password="sowmya1234$.",database="mydata")  
 my\_cursur=conn.cursor()  
 my\_cursur.execute("insert into employee values(%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\_idproofcomb.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()  
 self.clear\_data()  
 conn.close()

create a function called Add data it is used to add new employee details and saved according to the given columns in database.

def update\_data(self):  
 if (self.var\_dep.get()=="" or self.var\_email.get()=="" or self.var\_idproof.get()==""):  
 messagebox.showerror("Error","All Fields Are required")  
 else:  
 try:  
 update=messagebox.askyesno("Update","Are you sure update this employee data",parent=self.root)  
 if update>0:  
 conn=mysql.connector.connect(host="localhost",username="root",password="sowmya1234$.",database="mydata")  
 my\_cursur=conn.cursor()  
 my\_cursur.execute("update employee 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=%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\_idproofcomb.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()  
 self.clear\_data()  
 conn.close()

create a function called update data it is used to update the data from the database and save the data according to the given columns in database

def OnSelected(event): global mem\_id, UpdateWindow curItem = tree.focus()

contents = (tree.item(curItem)) selecteditem = contents['values'] mem\_id = selecteditem[0] FIRSTNAME.set("") LASTNAME.set("")

GENDER.set("")

AGE.set("")

FEEDBACK.set("")

FIRSTNAME.set(selecteditem[1]) LASTNAME.set(selecteditem[2]) AGE.set(selecteditem[4]) FEEDBACK.set(selecteditem[5]) UpdateWindow = Toplevel()

UpdateWindow.title("Customer Feedback Management System") width = 400

height = 300

screen\_width = root.winfo\_screenwidth() screen\_height = root.winfo\_screenheight()

x = ((screen\_width / 2) + 450) - (width / 2) y = ((screen\_height / 2) + 20) - (height / 2) UpdateWindow.resizable(0, 0)

UpdateWindow.geometry("%dx%d+%d+%d" % (width, height, x, y)) if 'NewWindow' in globals():

NewWindow.destroy()

create a function called onselected un pause the parameter called event and this function is used to arrange the given data the given columns in the database arrange the data by the order and the updated window you should also equal to that screen height and width if new window is in global it will close the window and the given data in updated window and the updated data we’re also saved in the following columns in database.

# manage Frame  
Manage\_frame=Frame(bg\_lbl,bd=2,relief=RIDGE,bg="white")  
Manage\_frame.place(x=15,y=10,width=1300,height=510)  
  
  
# left Frame  
upper\_frame=LabelFrame(Manage\_frame,bd=4,relief=RIDGE,padx=2,text="Employee Information",font=("times new roman",11,"bold"),fg="red",bg="white")  
upper\_frame.place(x=10,y=10,width=1300,height=270)

Frames it will form a title for the window on the radio button for male and female option

was created by radio button function and it will show in the window

# Employee class LabelFrame Information  
emp\_lbl\_frame=Frame(upper\_frame,bd=4,relief=RIDGE,padx=2,bg="white")  
emp\_lbl\_frame.place(x=0,y=10,width=1210,height=235)  
  
# labels entry  
# ID  
# department  
lbl\_dep=Label(emp\_lbl\_frame,text="Department",font=("arial",11,"bold"),bg="white")  
lbl\_dep.grid(row=0,column=0,padx=2,sticky=W)  
  
combo\_dep=ttk.Combobox(emp\_lbl\_frame,textvariable=self.var\_dep,font=("arial",12,"bold"),width=17,state="readonly")  
combo\_dep["value"]=("Select Department","HR","Software Engineer","Infrastracture","Manager")  
combo\_dep.current(0)  
combo\_dep.grid(row=0,column=1,padx=2,pady=10,sticky=W)

Labels is the column names in given futures like front style and size

# Button Frame  
button\_frame=Frame(upper\_frame,bd=2,relief=RIDGE,bg="white")  
button\_frame.place(x=1000,y=20,width=170,height=220)  
  
btn\_add=Button(button\_frame,text="Save",command=self.add\_data,font=("arial",15,"bold"),width=13,bg="blue",fg="white")  
btn\_add.grid(row=0,column=0,padx=1,pady=5)  
  
btn\_update=Button(button\_frame,text="Update",command=self.update\_data,font=("arial",15,"bold"),width=13,bg="blue",fg="white")  
btn\_update.grid(row=1,column=0,padx=1,pady=5)  
  
btn\_delete=Button(button\_frame,text="Delete",command=self.delete\_data,font=("arial",15,"bold"),width=13,bg="blue",fg="white")  
btn\_delete.grid(row=2,column=0,padx=1,pady=5)  
  
btn\_clear=Button(button\_frame,text="clear",command=self.clear\_data,font=("arial",15,"bold"),width=13,bg="blue",fg="white")  
btn\_clear.grid(row=3,column=0,padx=1,pady=5)

Buttons have the click function there are used to save the data and these buttons height and width are according to the size of window

# Delete  
def delete\_data(self):  
 if self.var\_idproof.get()=="":  
 messagebox.showerror("Error","All Fields Are required",parent=self.root)  
 else:  
 try:  
 Delete=messagebox.askyesno("Delete","Are sure delete this student",parent=self.root)  
 if Delete>0:  
 conn=mysql.connector.connect(host="localhost",username="root",password="sowmya1234$.",database="mydata")  
 my\_cursur=conn.cursor()  
 sql="delete from employee where id\_proof=%s"  
 value=(self.var\_idproof.get(),)  
 my\_cursur.execute(sql,value)  
 else:  
 if not Delete:  
 return  
 conn.commit()  
 self.fetch\_data()  
 conn.close()  
 messagebox.showinfo("Delete","Your Employee has been Deleted",parent=self.root)

create a function called delete data and this function is used to delete the data from the database. For delete the data from the database first we need to select the data from the database and press the delete button to delete the data

# ==============Employee Table and Scroll bar====================  
table\_frame=Frame(DataRightFrame,bd=4,relief=RIDGE)  
table\_frame.place(x=0,y=60,width=1200,height=100)  
  
scroll\_x=ttk.Scrollbar(table\_frame,orient=HORIZONTAL)  
scroll\_y=ttk.Scrollbar(table\_frame,orient=VERTICAL)  
self.employee\_table=ttk.Treeview(table\_frame,column=("dep","name","degi","email","address","married","dob","doj","idproofcomb","idproof","gender","phone","country","salary",),xscrollcommand=scroll\_x.set,yscrollcommand=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="Degignition")  
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("idproofcomb",text="ID Type")  
self.employee\_table.heading("idproof",text="ID Proof")  
self.employee\_table.heading("gender",text="Gender")  
self.employee\_table.heading("phone",text="Phone")  
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("idproofcomb",width=100)  
self.employee\_table.column("idproof",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()

Create a with the heading and hey good idea scroll bar for the table according to the height and width on the table

if \_\_name\_\_ == "\_\_main\_\_":  
 root=Tk()  
 obj=Employee(root)  
 root.mainloop()

Initialization

## Source code:

from tkinter import\*

from tkinter import ttk

from PIL import Image,ImageTk

import mysql.connector

from tkinter import messagebox

from tkinter import filedialog

from time import strftime

from datetime import datetime

import os

class Employee:

def \_\_init\_\_(self, root, upper\_frame=None):

self.root=root

self.root.geometry("1530x790+0+0")

self.root.title("EMPLOYEE MANAGEMENT SYSTEM")

# Variables

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\_idproofcomb=StringVar()

self.var\_idproof=StringVar()

self.var\_gender=StringVar()

self.var\_phone=StringVar()

self.var\_country=StringVar()

self.var\_salary=StringVar()

lbl\_title=Label(self.root,text="EMPLOYEE MANAGEMENT SYSTEM",font=("times new roman",37,"bold"),fg="darkblue",bg="white")

lbl\_title.place(x=0,y=0,width=1530,height=50)

img\_logo=Image.open(r"college\_images\emplogo.png")

img\_logo=img\_logo.resize((50,50),Image.ANTIALIAS)

self.photoimg\_logo=ImageTk.PhotoImage(img\_logo)

self.logo=Label(self.root,image=self.photoimg\_logo)

self.logo.place(x=270,y=0,width=50,height=50)

img\_frame=Frame(self.root,bd=2,relief=RIDGE,bg='white')

img\_frame.place(x=0,y=50,width=1530,height=140)

# 1st

img=Image.open(r"college\_images\emp5.jpg")

img=img.resize((540,140),Image.ANTIALIAS)

self.photoimg=ImageTk.PhotoImage(img)

self.btn\_1=Label(img\_frame,image=self.photoimg,)

self.btn\_1.place(x=0,y=0,width=540,height=140)

# 2st

img\_2=Image.open(r"college\_images\emp2.png")

img\_2=img\_2.resize((540,140),Image.ANTIALIAS)

self.photoimg\_2=ImageTk.PhotoImage(img\_2)

self.btn\_2=Label(img\_frame,image=self.photoimg\_2)

self.btn\_2.place(x=540,y=0,width=540,height=140)

# 3st

img\_3=Image.open(r"college\_images\emp4.jpg")

img\_3=img\_3.resize((540,140),Image.ANTIALIAS)

self.photoimg\_3=ImageTk.PhotoImage(img\_3)

self.btn\_3=Label(img\_frame,image=self.photoimg\_3)

self.btn\_3.place(x=1000,y=0,width=540,height=140)

# bg image

img\_4=Image.open(r"college\_images\university.jpg")

img\_4=img\_4.resize((1550,600),Image.ANTIALIAS)

self.photoimg\_4=ImageTk.PhotoImage(img\_4)

bg\_lbl=Label(self.root,image=self.photoimg\_4,bd=2,relief=RIDGE)

bg\_lbl.place(x=0,y=210,width=1530,height=580)

# =================time================================

# def time():

# string = strftime('%I:%M:%S %p')

# lbl.config(text = string)

# lbl.after(1000, time)

# lbl = Label(lbl\_title, font = ('times new roman',16, 'bold'),background = 'white',foreground = 'blue')

# lbl.place(x=0,y=(-15),width=120,height=50)

# time()

# manage Frame

Manage\_frame=Frame(bg\_lbl,bd=2,relief=RIDGE,bg="white")

Manage\_frame.place(x=15,y=10,width=1300,height=510)

# left Frame

upper\_frame=LabelFrame(Manage\_frame,bd=4,relief=RIDGE,padx=2,text="Employee Information",font=("times new roman",11,"bold"),fg="red",bg="white")

upper\_frame.place(x=10,y=10,width=1300,height=270)

# Employee class LabelFrame Information

emp\_lbl\_frame=Frame(upper\_frame,bd=4,relief=RIDGE,padx=2,bg="white")

emp\_lbl\_frame.place(x=0,y=10,width=1210,height=235)

# labels entry

# ID

# department

lbl\_dep=Label(emp\_lbl\_frame,text="Department",font=("arial",11,"bold"),bg="white")

lbl\_dep.grid(row=0,column=0,padx=2,sticky=W)

combo\_dep=ttk.Combobox(emp\_lbl\_frame,textvariable=self.var\_dep,font=("arial",12,"bold"),width=17,state="readonly")

combo\_dep["value"]=("Select Department","HR","Software Engineer","Infrastracture","Manager")

combo\_dep.current(0)

combo\_dep.grid(row=0,column=1,padx=2,pady=10,sticky=W)

# Name

lbl\_Name=Label(emp\_lbl\_frame,font=("arial",12,"bold"),text="Name:",bg="white")

lbl\_Name.grid(row=0,column=2,sticky=W,padx=2,pady=7)

txt\_name=ttk.Entry(emp\_lbl\_frame,textvariable=self.var\_name,width=22,font=("arial",11,"bold"))

txt\_name.grid(row=0,column=3,padx=2,pady=7)

# # lbl\_Designition

lbl\_Designition=Label(emp\_lbl\_frame,font=("arial",12,"bold"),text="Designition:",bg="white")

lbl\_Designition.grid(row=1,column=0,sticky=W,padx=2,pady=7)

txt\_Designition=ttk.Entry(emp\_lbl\_frame,textvariable=self.var\_designition,width=22,font=("arial",11,"bold"))

txt\_Designition.grid(row=1,column=1,sticky=W,padx=2,pady=7)

# Email

lbl\_email=Label(emp\_lbl\_frame,font=("arial",12,"bold"),text="Email:",bg="white")

lbl\_email.grid(row=1,column=2,sticky=W,padx=2,pady=7)

txt\_email=ttk.Entry(emp\_lbl\_frame,textvariable=self.var\_email,width=22,font=("arial",11,"bold"))

txt\_email.grid(row=1,column=3,padx=2,pady=7)

# Address

lbl\_adderss=Label(emp\_lbl\_frame,font=("arial",12,"bold"),text="Address:",bg="white")

lbl\_adderss.grid(row=2,column=0,sticky=W,padx=2,pady=7)

txt\_adderss=ttk.Entry(emp\_lbl\_frame,textvariable=self.var\_address,width=22,font=("arial",11,"bold"))

txt\_adderss.grid(row=2,column=1,padx=2,pady=7)

# Married

lbl\_merried\_status=Label(emp\_lbl\_frame,font=("arial",12,"bold"),text="Married Status:",bg="white")

lbl\_merried\_status.grid(row=2,column=2,sticky=W,padx=2,pady=7)

com\_txt\_married=ttk.Combobox(emp\_lbl\_frame,textvariable=self.var\_married,state="readonly",

font=("arial",12,"bold"),width=18)

com\_txt\_married['value']=("Married","Unmarried")

com\_txt\_married.current(0)

com\_txt\_married.grid(row=2,column=3,sticky=W,padx=2,pady=7)

# Dob

lbl\_dob=Label(emp\_lbl\_frame,font=("arial",12,"bold"),text="DOB:",bg="white")

lbl\_dob.grid(row=3,column=0,sticky=W,padx=2,pady=7)

txt\_dob=ttk.Entry(emp\_lbl\_frame,textvariable=self.var\_dob,width=22,font=("arial",11,"bold"))

txt\_dob.grid(row=3,column=1,padx=2,pady=7)

# Doj

lbl\_doj=Label(emp\_lbl\_frame,font=("arial",12,"bold"),text="DOJ:",bg="white")

lbl\_doj.grid(row=3,column=2,sticky=W,padx=2,pady=7)

txt\_doj=ttk.Entry(emp\_lbl\_frame,textvariable=self.var\_doj,width=22,font=("arial",11,"bold"))

txt\_doj.grid(row=3,column=3,padx=2,pady=7)

# Id Proof

# lbl\_id\_proof=Label(emp\_lbl\_frame,font=("arial",12,"bold"),text="ID Proof:",bg="white")

# lbl\_id\_proof.grid(row=4,column=0,sticky=W,padx=2,pady=7)

com\_txt\_proof=ttk.Combobox(emp\_lbl\_frame,textvariable=self.var\_idproofcomb,state="readonly",

font=("arial",12,"bold"),width=18)

com\_txt\_proof['value']=("Select ID Proof","PAN CARD","AADHAR CARD","DRIVING LICENCE")

com\_txt\_proof.current(0)

com\_txt\_proof.grid(row=4,column=0,sticky=W,padx=2,pady=7)

txt\_proof=ttk.Entry(emp\_lbl\_frame,textvariable=self.var\_idproof,width=22,font=("arial",11,"bold"))

txt\_proof.grid(row=4,column=1,padx=2,pady=7)

# gender

lbl\_gender=Label(emp\_lbl\_frame,font=("arial",12,"bold"),text="Gender:",bg="white")

lbl\_gender.grid(row=4,column=2,sticky=W,padx=2,pady=7)

com\_txt\_gender=ttk.Combobox(emp\_lbl\_frame,textvariable=self.var\_gender,state="readonly",

font=("arial",12,"bold"),width=18)

com\_txt\_gender['value']=("Male","Female","Other")

com\_txt\_gender.current(0)

com\_txt\_gender.grid(row=4,column=3,sticky=W,padx=2,pady=7)

# phone

lbl\_phone=Label(emp\_lbl\_frame,font=("arial",12,"bold"),text="Phone No:",bg="white")

lbl\_phone.grid(row=0,column=4,sticky=W,padx=2,pady=7)

txt\_phone=ttk.Entry(emp\_lbl\_frame,textvariable=self.var\_phone,width=22,font=("arial",11,"bold"))

txt\_phone.grid(row=0,column=5,padx=2,pady=7)

# # country

lbl\_country=Label(emp\_lbl\_frame,font=("arial",12,"bold"),text="Country:",bg="white")

lbl\_country.grid(row=1,column=4,sticky=W,padx=2,pady=7)

txt\_country=ttk.Entry(emp\_lbl\_frame,textvariable=self.var\_country,width=22,font=("arial",11,"bold"))

txt\_country.grid(row=1,column=5,padx=2,pady=7)

# # CTC

lbl\_ctc=Label(emp\_lbl\_frame,font=("arial",12,"bold"),text="Salary(CTC):",bg="white")

lbl\_ctc.grid(row=2,column=4,sticky=W,padx=2,pady=7)

txt\_ctc=ttk.Entry(emp\_lbl\_frame,textvariable=self.var\_salary,width=22,font=("arial",11,"bold"))

txt\_ctc.grid(row=2,column=5,padx=2,pady=7)

# stayhome=Label(emp\_lbl\_frame,text="STAY HOME STAY SAFE",font=("times new roman",30,"bold"),fg="red",bg="white")

# stayhome.place(x=780,y=0,width=670,height=30)

# Button Frame

button\_frame=Frame(upper\_frame,bd=2,relief=RIDGE,bg="white")

button\_frame.place(x=1000,y=20,width=170,height=220)

btn\_add=Button(button\_frame,text="Save",command=self.add\_data,font=("arial",15,"bold"),width=13,bg="blue",fg="white")

btn\_add.grid(row=0,column=0,padx=1,pady=5)

btn\_update=Button(button\_frame,text="Update",command=self.update\_data,font=("arial",15,"bold"),width=13,bg="blue",fg="white")

btn\_update.grid(row=1,column=0,padx=1,pady=5)

btn\_delete=Button(button\_frame,text="Delete",command=self.delete\_data,font=("arial",15,"bold"),width=13,bg="blue",fg="white")

btn\_delete.grid(row=2,column=0,padx=1,pady=5)

btn\_clear=Button(button\_frame,text="clear",command=self.clear\_data,font=("arial",15,"bold"),width=13,bg="blue",fg="white")

btn\_clear.grid(row=3,column=0,padx=1,pady=5)

# Down Frame

DataRightFrame=LabelFrame(Manage\_frame,bd=4,relief=RIDGE,padx=2,text="Employee Information",font=("times new roman",11,"bold"),fg="darkblue",bg="white")

DataRightFrame.place(x=10,y=280,width=1300,height=200)

# Down Frame

Search\_Frame=LabelFrame(DataRightFrame,bd=4,relief=RIDGE,padx=2,text="Search Employee Information",font=("times new roman",11,"bold"),fg="darkblue",bg="white")

Search\_Frame.place(x=0,y=0,width=1200,height=60)

search\_by=Label(Search\_Frame,font=("arial",11,"bold"),text="Search By:",fg="white",bg="red")

search\_by.grid(row=0,column=0,sticky=W,padx=5)

# search

self.var\_com\_search=StringVar()

com\_txt\_search=ttk.Combobox(Search\_Frame,textvariable=self.var\_com\_search,state="readonly",

font=("arial",12,"bold"),width=18)

com\_txt\_search['value']=("Select Option","Phone","id\_proof")

com\_txt\_search.current(0)

com\_txt\_search.grid(row=0,column=1,sticky=W,padx=5)

self.var\_search=StringVar()

txt\_search=ttk.Entry(Search\_Frame,textvariable=self.var\_search,width=22,font=("arial",11,"bold"))

txt\_search.grid(row=0,column=2,padx=5)

btn\_\_search=Button(Search\_Frame,command=self.search\_data,text="Search",font=("arial",11,"bold"),width=14,bg="blue",fg="white")

btn\_\_search.grid(row=0,column=3,padx=5)

btn\_ShowAll=Button(Search\_Frame,command=self.fetch\_data,text="Show All",font=("arial",11,"bold"),width=14,bg="blue",fg="white")

btn\_ShowAll.grid(row=0,column=4,padx=5)

stayhome=Label(Search\_Frame,text=" ",font=("times new roman",30,"bold"),fg="red",bg="white")

stayhome.place(x=780,y=0,width=600,height=30)

img\_logo\_mask=Image.open(r"college\_images\mask.jpg")

img\_logo\_mask=img\_logo\_mask.resize((50,50),Image.ANTIALIAS)

self.photoimg\_logo\_mask=ImageTk.PhotoImage(img\_logo\_mask)

# ==============Student Table and Scroll bar====================

table\_frame=Frame(DataRightFrame,bd=4,relief=RIDGE)

table\_frame.place(x=0,y=60,width=1200,height=100)

scroll\_x=ttk.Scrollbar(table\_frame,orient=HORIZONTAL)

scroll\_y=ttk.Scrollbar(table\_frame,orient=VERTICAL)

self.employee\_table=ttk.Treeview(table\_frame,column=("dep","name","degi","email","address","married","dob","doj","idproofcomb","idproof","gender","phone","country","salary",),xscrollcommand=scroll\_x.set,yscrollcommand=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="Degignition")

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("idproofcomb",text="ID Type")

self.employee\_table.heading("idproof",text="ID Proof")

self.employee\_table.heading("gender",text="Gender")

self.employee\_table.heading("phone",text="Phone")

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("idproofcomb",width=100)

self.employee\_table.column("idproof",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()

def add\_data(self):

if (self.var\_dep.get()=="" or self.var\_email.get()=="" or self.var\_idproof.get()==""):

messagebox.showerror("Error","All Fields Are required")

else:

try:

conn=mysql.connector.connect(host="localhost",username="root",password="sowmya1234$.",database="mydata")

my\_cursur=conn.cursor()

my\_cursur.execute("insert into employee values(%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\_idproofcomb.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()

self.clear\_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 Function

def fetch\_data(self):

conn=mysql.connector.connect(host="localhost",username="root",password="sowmya1234$.",database="mydata")

my\_cursur=conn.cursor()

my\_cursur.execute("select \* from employee")

data=my\_cursur.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\_idproofcomb.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()=="" or self.var\_idproof.get()==""):

messagebox.showerror("Error","All Fields Are required")

else:

try:

update=messagebox.askyesno("Update","Are you sure update this employee data",parent=self.root)

if update>0:

conn=mysql.connector.connect(host="localhost",username="root",password="sowmya1234$.",database="mydata")

my\_cursur=conn.cursor()

my\_cursur.execute("update employee 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=%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\_idproofcomb.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()

self.clear\_data()

conn.close()

messagebox.showinfo("Success","Employee successfully updaded",parent=self.root)

except Exception as es:

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

# Delete

def delete\_data(self):

if self.var\_idproof.get()=="":

messagebox.showerror("Error","All Fields Are required",parent=self.root)

else:

try:

Delete=messagebox.askyesno("Delete","Are sure delete this student",parent=self.root)

if Delete>0:

conn=mysql.connector.connect(host="localhost",username="root",password="sowmya1234$.",database="mydata")

my\_cursur=conn.cursor()

sql="delete from employee where id\_proof=%s"

value=(self.var\_idproof.get(),)

my\_cursur.execute(sql,value)

else:

if not Delete:

return

conn.commit()

self.fetch\_data()

conn.close()

messagebox.showinfo("Delete","Your Employee has been Deleted",parent=self.root)

except Exception as es:

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

# clear

def clear\_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("Married")

self.var\_dob.set("")

self.var\_doj.set("")

self.var\_idproofcomb.set("Select ID Proof")

self.var\_idproof.set("")

self.var\_gender.set("Male")

self.var\_phone.set("")

self.var\_country.set("")

self.var\_salary.set("")

# search data

def search\_data(self):

if self.var\_com\_search.get()=="" or self.var\_search.get()=="":

messagebox.showerror("Error","Plaese select option")

else:

try:

conn=mysql.connector.connect(host='localhost',username='root',password='sowmya1234$.',database='mydata')

my\_cursor=conn.cursor()

my\_cursor.execute("select \* from employee 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(

## SCREEN SHOTS:

