**DBMS MINI PROJECT:**

**Employee Database Management System(EDMS)**

**Submitted by:**

**Shrivathsan S (312322205165)**

**Vishnu Priyan K(312322205194)**

**Of**

**BACHELOR OF TECHNOLOGY**

**In**

**INFORMATION TECHNOLOGY**

****

**St. JOSEPH’S COLLEGE OF ENGINEERING**

**(An Autonomous Institution)**

**St. Joseph’s Group of Institutions**

**OMR, Chennai 600 119**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| CHAPTER | TITLE | PAGE NO. |
| 1. | ABSTRACT | 3 |
| 2. | INTRODUCTION | 3 |
| 3. | PROGRAM | 4 – 8 |
| 4. | WORKFLOW | 9-10 |
| 5. | RESULT AND OUTPUT | 10-13 |
| 6. | CONCLUSION | 14 |

**Abstract:**

A Python-based Employee Database Management System (DBMS) built with MySQL and tkinter. The system facilitates CRUD (Create, Read, Update, Delete) operations for employee records. Users can add new employees, view existing records, update employee details, and delete records. The GUI, developed using tkinter, provides an intuitive interface for interaction. MySQL is utilized as the backend database for storing employee information. The system offers a streamlined approach to managing employee data, enhancing efficiency in record-keeping and manipulation.

**Introduction:**

The Python-based Employee Database Management System (DBMS) is a user-friendly solution designed to streamline the management of employee records. Leveraging the power of MySQL as the backend database and tkinter for the graphical user interface (GUI), the system empowers users to perform CRUD (Create, Read, Update, Delete) operations effortlessly. With features for adding, viewing, updating, and deleting employee records, it offers a seamless experience for organizing and manipulating employee data. This project aims to enhance efficiency in record-keeping tasks, providing a reliable and intuitive platform for managing employee information in diverse organizational settings.

**Source Code:**

import mysql.connector

import tkinter as tk

from tkinter import messagebox

class EmployeeDBGUI:

def \_\_init\_\_(self, host, user, password, database):

self.conn=mysql.connector.connect(host=host,user=user,password=password, database=database)

self.cur = self.conn.cursor()

self.create\_table()

self.root = tk.Tk()

self.root.title("Employee Database")

self.label\_name = tk.Label(self.root, text="Name:")

self.label\_name.grid(row=0, column=0)

self.entry\_name = tk.Entry(self.root)

self.entry\_name.grid(row=0, column=1)

self.label\_age = tk.Label(self.root, text="Age:")

self.label\_age.grid(row=1, column=0)

self.entry\_age = tk.Entry(self.root)

self.entry\_age.grid(row=1, column=1)

self.label\_department = tk.Label(self.root, text="Department:")

self.label\_department.grid(row=2, column=0)

self.entry\_department = tk.Entry(self.root)

self.entry\_department.grid(row=2, column=1)

self.button\_add = tk.Button(self.root, text="Add Employee", command=self.add\_employee)

self.button\_add.grid(row=3, column=0, columnspan=2)

self.button\_view = tk.Button(self.root, text="View Employees", command=self.view\_employees)

self.button\_view.grid(row=4, column=0, columnspan=2)

self.label\_update\_id = tk.Label(self.root, text="Employee ID to Update:")

self.label\_update\_id.grid(row=5, column=0)

self.entry\_update\_id = tk.Entry(self.root)

self.entry\_update\_id.grid(row=5, column=1)

self.button\_update = tk.Button(self.root, text="Update Employee", command=self.update\_employee)

self.button\_update.grid(row=6, column=0, columnspan=2)

self.label\_delete\_id = tk.Label(self.root, text="Employee ID to Delete:")

self.label\_delete\_id.grid(row=7, column=0)

self.entry\_delete\_id = tk.Entry(self.root)

self.entry\_delete\_id.grid(row=7, column=1)

self.button\_delete = tk.Button(self.root, text="Delete Employee", command=self.delete\_employee)

self.button\_delete.grid(row=8, column=0, columnspan=2)

self.root.mainloop()

def create\_table(self):

self.cur.execute('''CREATE TABLE IF NOT EXISTS employees (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255),

age INT,

department VARCHAR(255))''')

self.conn.commit()

def add\_employee(self):

name = self.entry\_name.get()

age = int(self.entry\_age.get())

department = self.entry\_department.get()

sql = "INSERT INTO employees (name, age, department) VALUES (%s, %s, %s)"

values = (name, age, department)

self.cur.execute(sql, values)

self.conn.commit()

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

def view\_employees(self):

self.cur.execute("SELECT \* FROM employees")

rows = self.cur.fetchall()

if not rows:

messagebox.showinfo("Info", "No employees found.")

else:

for row in rows:

print(row)

self.conn.commit()

def update\_employee(self):

id = int(self.entry\_update\_id.get())

name = self.entry\_name.get()

age = int(self.entry\_age.get())

department = self.entry\_department.get()

sql = "UPDATE employees SET name=%s, age=%s, department=%s WHERE id=%s"

values = (name, age, department, id)

self.cur.execute(sql, values)

self.conn.commit()

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

def delete\_employee(self):

id = int(self.entry\_delete\_id.get())

sql = "DELETE FROM employees WHERE id=%s"

values = (id,)

self.cur.execute(sql, values)

self.conn.commit()

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

def \_\_del\_\_(self):

self.conn.close()

# Example Usage

if \_\_name\_\_ == "\_\_main\_\_":

db = EmployeeDBGUI(host='localhost', user='root', password='Shrivathsan007', database='dbp')

**Workflow:**

1. User Interaction: The system begins with user interaction through the graphical user interface (GUI). Users input employee details such as name, age, and department via text entry fields provided by the GUI. These details are essential for various operations like adding new employees, viewing existing records, updating employee details, or deleting records.

2. Database Interaction: Upon receiving user input, the system interacts with the MySQL database to execute corresponding SQL queries. These queries perform CRUD (Create, Read, Update, Delete) operations on the database. For instance, an INSERT query adds a new employee record, a SELECT query retrieves existing records, an UPDATE query modifies employee details, and a DELETE query removes employee records.

3. Feedback Mechanism: After each database operation, the system provides feedback to the user through message boxes or notifications displayed in the GUI. This feedback communicates the success or failure of the operation and any relevant information, ensuring transparency and user awareness.

4. Iteration: The workflow iterates continuously as users navigate through the GUI, performing multiple operations as needed. Users can seamlessly manage employee data, refining and updating records to maintain an accurate and up-to-date database.

**Tech stack**:

1. Programming Language: Python

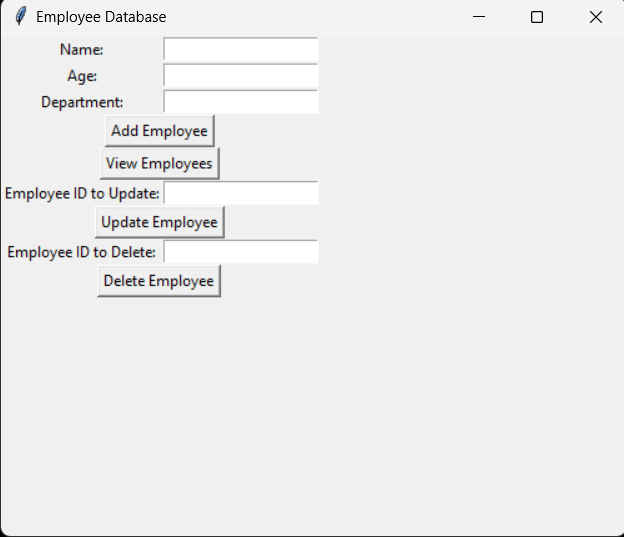
2. GUI Framework: tkinter (Python's standard GUI library)

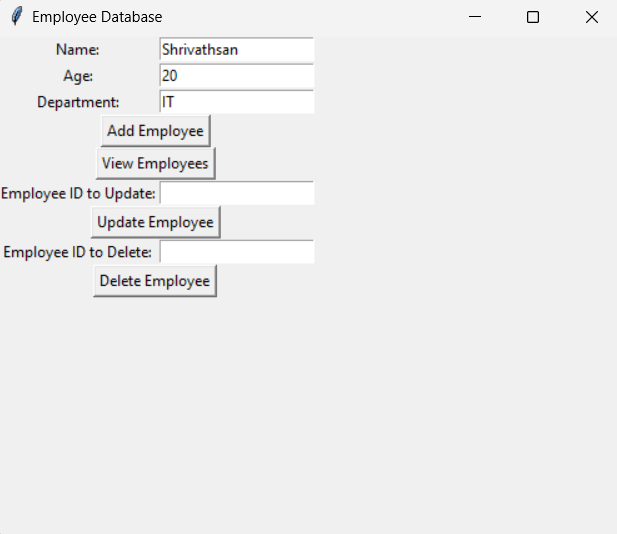
3. Database Management System: MySQL

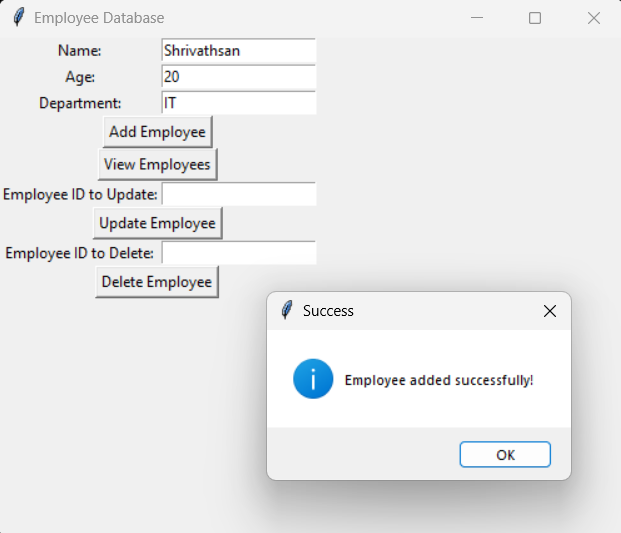
4. Database Connectivity: mysql-connector-python (Python module for connecting to MySQL databases)

**Output and Result:**

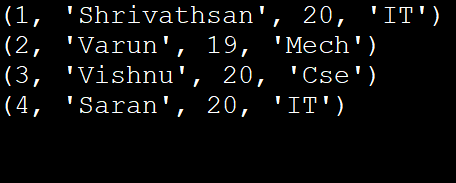
The Employee Database Management System simplifies user input through a user-friendly GUI, allowing seamless interaction with the MySQL database for CRUD operations. Prompt feedback ensures transparency and user awareness. This system empowers efficient management of employee records, enabling users to add, view, update, and delete entries with ease. The combination of intuitive interface and robust database functionality ensures a streamlined and effective management process for employee data.



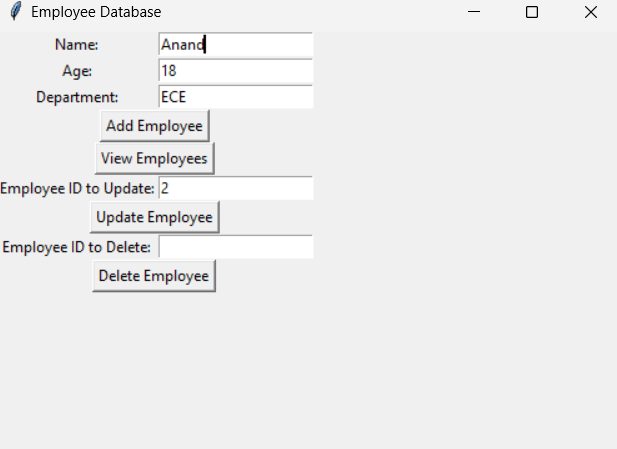


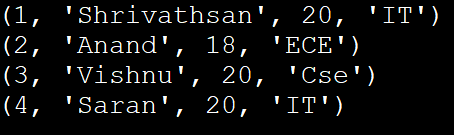


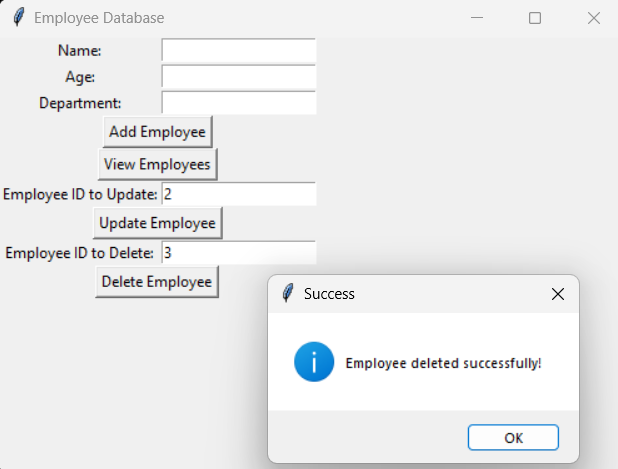
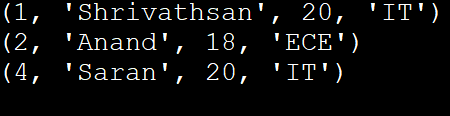
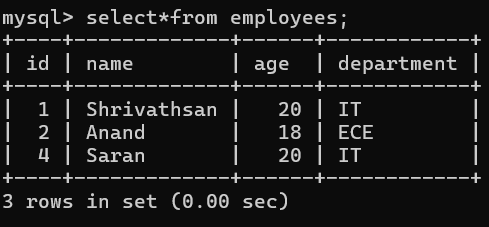
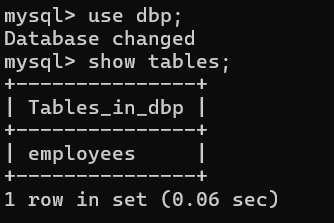
* Here the records have been added to the table



* Here the record 2 has been updated successfully





* Deletion:
* Mysql Table:

**Conclusion:**

The Employee Database Management System presents an efficient solution for managing employee records with ease. By leveraging the GUI interface and MySQL database interaction, users can seamlessly add, view, update, and delete employee details. The system's feedback mechanism ensures transparency and user awareness throughout the process. Overall, this project offers a streamlined approach to employee data management, enhancing organizational efficiency and facilitating effective decision-making.

Thank You