**Project Title**

Design a database application using python GUI to modify specified record of a teacher using teacher id, with the help of database and display the modified record (teacher id, name, subject, dept., years of

experience and salary)

**Group Members:-**

Hanumantha Kamble (61)

Jay Pinage (36)

Yash Sharma (60)

Krishna Rajbhar (38)

Manas joshi (23)

**Project Description:-**

Our project was about making a teachers management using Python GUI Tkinter and with MySQL Database

In this project we have added several features such adding teachers data with Modifications of teachers data using ID, Deletion of teachers data using ID and also we have created a grid view for displaying teachers data in Python GUI application with fetching the data from MySQL database.

After completing this module we have connected theses fields with MySQL database Where we added the mysql field such as

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100),

subject VARCHAR(100),

dept VARCHAR(100),

years\_of\_experience INT,

salary DECIMAL(10, 2)

We have used MySQL-connector-python module for connecting the python GUI with MySQL using user and password, that code goes like

def connect\_database():

return mysql.connector.connect(

host="localhost",

user="root",

password="12345@abc123",

database="techboys"

**Code Snippets:-**

import tkinter as tk

from tkinter import messagebox, ttk

import mysql.connector

def connect\_database():

return mysql.connector.connect(

host="localhost",

user="root",

password="techboyhk0005@",

database="school\_db"

)

def add\_teacher():

name = name\_entry.get()

subject = subject\_entry.get()

dept = dept\_entry.get()

years\_of\_experience = years\_entry.get()

salary = salary\_entry.get()

if not (name and subject and dept and years\_of\_experience and salary):

messagebox.showerror("Input Error", "Please fill all fields")

return

try:

years\_of\_experience = int(years\_of\_experience)

salary = float(salary)

except ValueError:

messagebox.showerror("Input Error", "Years of experience must be an integer and salary must be a number")

return

conn = connect\_database()

cursor = conn.cursor()

cursor.execute('''

INSERT INTO teachers (name, subject, dept, years\_of\_experience, salary)

VALUES (%s, %s, %s, %s, %s)

''', (name, subject, dept, years\_of\_experience, salary))

conn.commit()

conn.close()

messagebox.showinfo("Success", "Teacher added successfully")

display\_teachers()

clear\_fields()

def modify\_teacher():

teacher\_id = id\_entry.get()

name = name\_entry.get()

subject = subject\_entry.get()

dept = dept\_entry.get()

years\_of\_experience = years\_entry.get()

salary = salary\_entry.get()

if not (teacher\_id and name and subject and dept and years\_of\_experience and salary):

messagebox.showerror("Input Error", "Please fill all fields")

return

try:

years\_of\_experience = int(years\_of\_experience)

salary = float(salary)

except ValueError:

messagebox.showerror("Input Error", "Years of experience must be an integer and salary must be a number")

return

conn = connect\_database()

cursor = conn.cursor()

cursor.execute('''

UPDATE teachers

SET name = %s, subject = %s, dept = %s, years\_of\_experience = %s, salary = %s

WHERE id = %s

''', (name, subject, dept, years\_of\_experience, salary, teacher\_id))

conn.commit()

if cursor.rowcount == 0:

messagebox.showerror("Update Error", "No record found with the given ID")

else:

messagebox.showinfo("Success", "Record updated successfully")

conn.close()

display\_teachers()

clear\_fields()

def delete\_teacher():

teacher\_id = id\_entry.get()

if not teacher\_id:

messagebox.showerror("Input Error", "Please provide an ID to delete")

return

conn = connect\_database()

cursor = conn.cursor()

cursor.execute('DELETE FROM teachers WHERE id = %s', (teacher\_id,))

conn.commit()

if cursor.rowcount == 0:

messagebox.showerror("Delete Error", "No record found with the given ID")

else:

messagebox.showinfo("Success", "Teacher deleted successfully")

conn.close()

display\_teachers()

clear\_fields()

def display\_teachers():

for row in tree.get\_children():

tree.delete(row)

conn = connect\_database()

cursor = conn.cursor()

cursor.execute('SELECT \* FROM teachers')

for row in cursor.fetchall():

tree.insert('', tk.END, values=row)

conn.close()

def clear\_fields():

id\_entry.delete(0, tk.END)

name\_entry.delete(0, tk.END)

subject\_entry.delete(0, tk.END)

dept\_entry.delete(0, tk.END)

years\_entry.delete(0, tk.END)

salary\_entry.delete(0, tk.END)

root = tk.Tk()

root.title("Teacher Information")

tk.Label(root, text="ID").grid(row=0, column=0)

id\_entry = tk.Entry(root)

id\_entry.grid(row=0, column=1)

tk.Label(root, text="Name:").grid(row=1, column=0)

name\_entry = tk.Entry(root)

name\_entry.grid(row=1, column=1)

tk.Label(root, text="Subject:").grid(row=2, column=0)

subject\_entry = tk.Entry(root)

subject\_entry.grid(row=2, column=1)

tk.Label(root, text="Dept:").grid(row=3, column=0)

dept\_entry = tk.Entry(root)

dept\_entry.grid(row=3, column=1)

tk.Label(root, text="Years of Experience:").grid(row=4, column=0)

years\_entry = tk.Entry(root)

years\_entry.grid(row=4, column=1)

tk.Label(root, text="Salary:").grid(row=5, column=0)

salary\_entry = tk.Entry(root)

salary\_entry.grid(row=5, column=1)

# Buttons in a single horizontal row

button\_frame = tk.Frame(root)

button\_frame.grid(row=6, column=0, columnspan=2, pady=10)

tk.Button(button\_frame, text="Add Teacher", command=add\_teacher).pack(side=tk.LEFT, padx=5)

tk.Button(button\_frame, text="Modify Teacher", command=modify\_teacher).pack(side=tk.LEFT, padx=5)

tk.Button(button\_frame, text="Delete Teacher", command=delete\_teacher).pack(side=tk.LEFT, padx=5)

tk.Button(button\_frame, text="Clear Fields", command=clear\_fields).pack(side=tk.LEFT, padx=5)

columns = ("id", "name", "subject", "dept", "years\_of\_experience", "salary")

tree = ttk.Treeview(root, columns=columns, show='headings')

for col in columns:

tree.heading(col, text=col.capitalize())

tree.column(col, width=100)

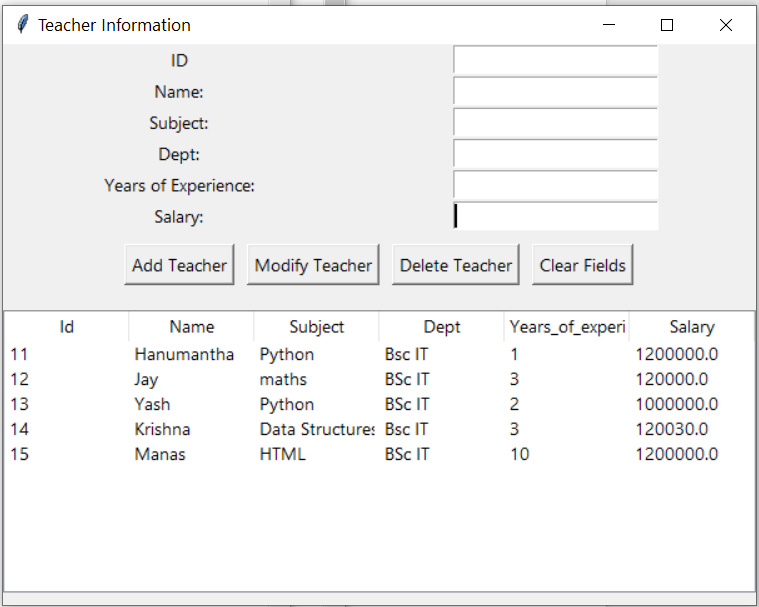
tree.grid(row=7, column=0, columnspan=2, pady=10)

display\_teachers()

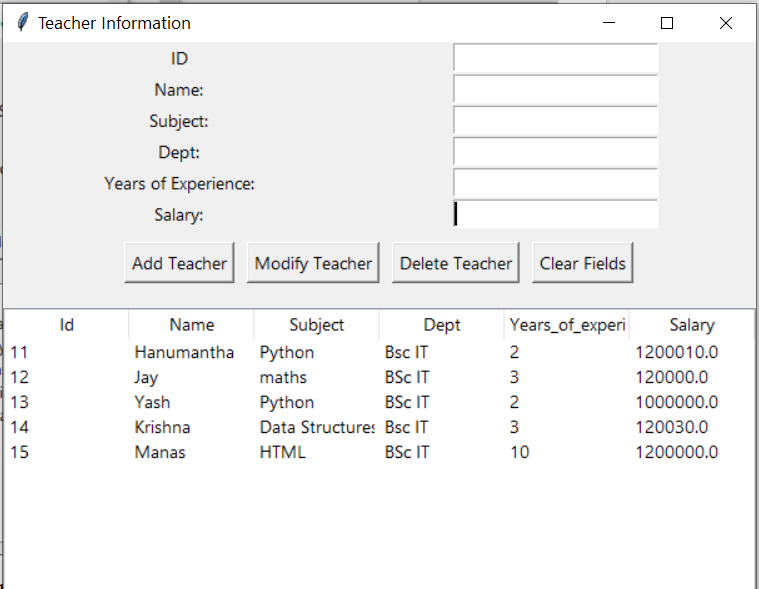
root.mainloop

**Output Snippets:-**

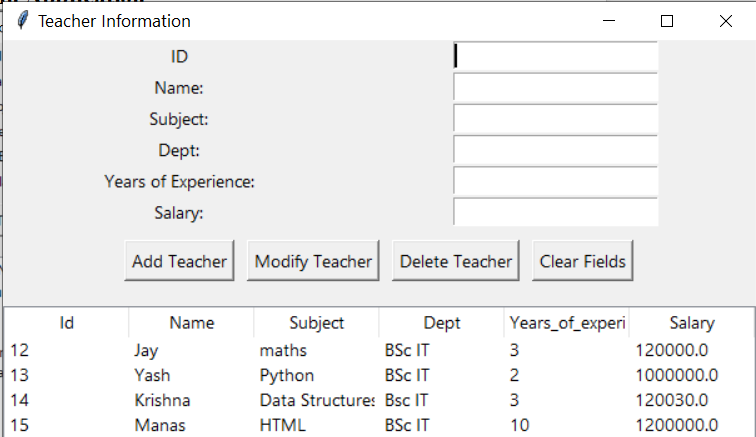
Main Application after running in Python IDLE



Modification in Application



Deletion in Application



**Conclusion:-**

This Python program using the Tkinter GUI toolkit and MySQL as the database backend provides a simple system for managing teacher information. Key features of the program include:

* Database Connection: It connects to a MySQL database (school\_db) on a local machine using the MySQL.connector package. The user credentials for accessing the database are customization.
* Add, Modify, Delete Teachers: The user can input a teacher's details such as name, subject, department, years of experience, and salary. The program allows adding new teachers, modifying existing teacher records by ID, and deleting records.
* Display Data: Teacher records are displayed in a tree view showing all the data fetched from the MySQL database. The view is updated after each action (add, modify, delete) to reflect the changes.
* Field Validation: Before submitting any data, the program checks that all fields are filled and that the "years of experience" is an integer and "salary" is a float, ensuring that only valid data gets processed.

**Git-hub Documentation URL:-**

https://github.com/jai99901/Mini-Project-of-Python/