Database Connectivity: Write a program to implement MongoDB database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.)

ANS=

To implement MongoDB database connectivity with a front-end language, we can use Python with a GUI library like Tkinter. Below is a complete example of a program that connects to a MongoDB database, allowing you to perform basic CRUD (Create, Read, Update, Delete) operations on a sample Employees collection.

Step 1: Install Required Packages

You need to install the pymongo package to connect to a MongoDB database. You can install it using pip:

bash

Copy code

pip install pymongo

Step 2: Set Up MongoDB

Before we create the program, ensure that you have a MongoDB database set up. If you don't have MongoDB installed, you can use MongoDB Atlas, a cloud-based database service.

1. **Create a Database**: Name it company.
2. **Create a Collection**: Name it employees.

You can do this using the MongoDB shell or GUI tools like MongoDB Compass.

Step 3: Create the Python Program

Below is a complete Python program that implements a simple GUI for adding, deleting, and editing employees in the employees collection.

python

Copy code

```
import tkinter as tk

from tkinter import messagebox

from pymongo import MongoClient

from bson.objectid import ObjectId


class DatabaseApp:

  def __init__(self, root):

    self.root = root

    self.root.title("Employee Database")


    # MongoDB Connection
```

```
        self.client = MongoClient("mongodb://localhost:27017/")  # Change if using MongoDB Atlas
        self.db = self.client["company"]
        self.collection = self.db["employees"]

        # GUI Components
        self.create_widgets()

    def create_widgets(self):
        self.emp_id_label = tk.Label(self.root, text="Employee ID:")
        self.emp_id_label.grid(row=0, column=0)
        self.emp_id_entry = tk.Entry(self.root)
        self.emp_id_entry.grid(row=0, column=1)

        self.emp_name_label = tk.Label(self.root, text="Employee Name:")
        self.emp_name_label.grid(row=1, column=0)
        self.emp_name_entry = tk.Entry(self.root)
        self.emp_name_entry.grid(row=1, column=1)

        self.emp_salary_label = tk.Label(self.root, text="Employee Salary:")
        self.emp_salary_label.grid(row=2, column=0)
        self.emp_salary_entry = tk.Entry(self.root)
        self.emp_salary_entry.grid(row=2, column=1)

        self.add_button = tk.Button(self.root, text="Add", command=self.add_employee)
        self.add_button.grid(row=3, column=0)

        self.delete_button = tk.Button(self.root, text="Delete", command=self.delete_employee)
        self.delete_button.grid(row=3, column=1)

        self.edit_button = tk.Button(self.root, text="Edit", command=self.edit_employee)
        self.edit_button.grid(row=3, column=2)
```

```
        self.show_button = tk.Button(self.root, text="Show Employees",
command=self.show_employees)
        self.show_button.grid(row=4, column=0, columnspan=3)

    def add_employee(self):
        emp_name = self.emp_name_entry.get()
        emp_salary = self.emp_salary_entry.get()
        if emp_name and emp_salary:
            try:
                emp_salary = float(emp_salary)
                self.collection.insert_one({"name": emp_name, "salary": emp_salary})
                messagebox.showinfo("Success", "Employee added successfully")
                self.clear_entries()
            except Exception as e:
                messagebox.showerror("Error", str(e))
        else:
            messagebox.showwarning("Input Error", "Please fill all fields.")

    def delete_employee(self):
        emp_id = self.emp_id_entry.get()
        if emp_id:
            try:
                self.collection.delete_one({"_id": ObjectId(emp_id)})
                messagebox.showinfo("Success", "Employee deleted successfully")
                self.clear_entries()
            except Exception as e:
                messagebox.showerror("Error", str(e))
        else:
            messagebox.showwarning("Input Error", "Please enter Employee ID.")
```

```
  def edit_employee(self):
    emp_id = self.emp_id_entry.get()
    emp_name = self.emp_name_entry.get()
    emp_salary = self.emp_salary_entry.get()
    if emp_id and emp_name and emp_salary:
      try:
        emp_salary = float(emp_salary)
        self.collection.update_one(
          {"_id": ObjectId(emp_id)},
          {"$set": {"name": emp_name, "salary": emp_salary}}
        )
        messagebox.showinfo("Success", "Employee updated successfully")
        self.clear_entries()
      except Exception as e:
        messagebox.showerror("Error", str(e))
    else:
      messagebox.showwarning("Input Error", "Please fill all fields.")

  def show_employees(self):
    employees = self.collection.find()
    employee_list = ""
    for emp in employees:
      employee_list += f"ID: {emp['_id']}, Name: {emp['name']}, Salary: {emp['salary']}\n"
    messagebox.showinfo("Employees", employee_list if employee_list else "No employees
found.")

  def clear_entries(self):
    self.emp_id_entry.delete(0, tk.END)
    self.emp_name_entry.delete(0, tk.END)
    self.emp_salary_entry.delete(0, tk.END)
```

```
if __name__ == "__main__":
    root = tk.Tk()
    app = DatabaseApp(root)
    root.mainloop()
```

Step 4: Run the Program

1. **Configure Database Connection:**
    - If you're using MongoDB Atlas, replace the connection string in the MongoClient constructor with your connection string.
2. **Run the Program:**
    - Execute the Python script. A GUI window will appear where you can add, delete, or edit employee records.

Summary

This program provides a simple GUI to interact with a MongoDB database for managing employee records. You can perform the following operations:

- **Add Employee**: Enter the employee's name and salary, and click "Add."
- **Delete Employee**: Enter the employee ID and click "Delete."
- **Edit Employee**: Enter the employee ID, name, and salary, and click "Edit."
- **Show Employees**: Click "Show Employees" to see a list of all employees in the database.

Additional Considerations

- **Error Handling**: Ensure proper error handling for cases like duplicate entries or invalid inputs.
- **UI Enhancements**: You can further enhance the UI and features based on your requirements.
- **MongoDB Cloud**: If you're using MongoDB Atlas, make sure to whitelist your IP address in the security settings to allow connections.

This simple application should help you understand how to interact with a MongoDB database from a Python GUI application.