# Artificial Intelligence
# Group C - Mini Project

Name: Faizan Kalkoti
Roll no: CO3016
TE Comp

AIM: Implement any of the following Expert System

1. Information management
2. Hospitals and medical facilities
3. Help desks management
4. Employee performance evaluation
5. Stock market trading
6. Airline scheduling and cargo schedules

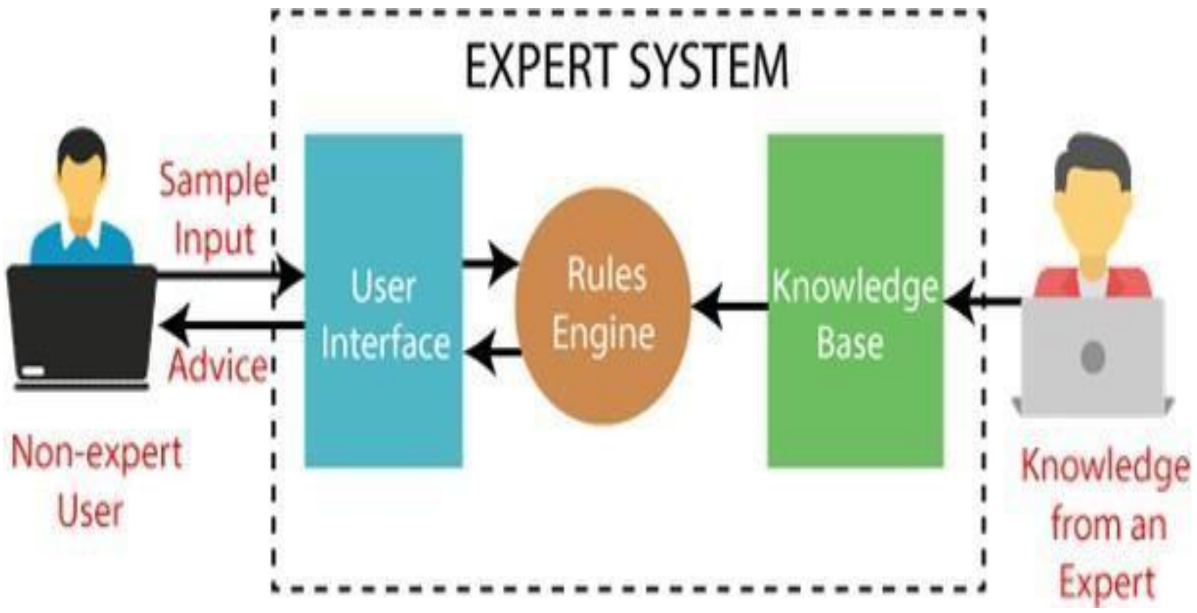
## What is an Expert System?

An expert system is a computer program that is designed to solve complex problems and to provide decision-making ability like a human expert. It performs this by extracting knowledge from its knowledge base using the reasoning and inference rules according to the user queries.

The expert system is a part of AI, and the first ES was developed in the year 1970, which was the first successful approach of artificial intelligence. It solves the most complex issue as an expert by extracting the knowledge stored in its knowledge base. The system helps in decision making for complex problems using both facts and heuristics like a human expert. It is called so because it contains the expert knowledge of a specific domain and can solve any complex problem of that domain. These systems are designed for a specific domain, such as medicine, science, etc.

The performance of an expert system is based on the expert's knowledge stored in its knowledge base. The more knowledge stored in the KB, the more that system improves its performance. One of the common examples of an ES is a suggestion of spelling errors while typing in the Google search box.

Below is the block diagram that represents the working of an expert system:



**Introduction:**

The Employee Performance Evaluation program aims to provide a convenient and efficient solution for managing employee performance evaluations in organizations. The project's objectives include automating the process of recording, retrieving, and analyzing employee performance data, ultimately facilitating performance assessment and decision-making.

Employee performance evaluation plays a crucial role in organizations as it enables employers to assess individual contributions, identify areas for improvement, and make informed decisions regarding promotions, rewards, and training opportunities. By implementing an automated system, organizations can streamline the evaluation process, enhance data accuracy, and save time and effort.

The program's purpose is to allow users to easily add employee evaluations, search for specific evaluations based on employee ID or other criteria, remove evaluations when needed, and generate comprehensive performance reports. By providing these functionalities, the program simplifies the management of employee performance data and enables quick access to relevant information.

**System Design and Implementation:**

The program follows a modular and object-oriented design approach. It consists of a class called Employee Performance Evaluation that encapsulates the functionality related to managing employee evaluations. The program utilizes file handling techniques to store and retrieve employee performance data in a text file named 'employee.txt'.

The data structure employed in the program is a simple text file where each line represents an employee evaluation. The data is stored in a comma-separated format, with the employee ID and performance rating being recorded for each evaluation.

The EmployeePerformanceEvaluator class incorporates methods and functions to handle various operations. These include the add_employee() method to add new evaluations, the search_employee() method to search for specific evaluations based on employee ID, the remove_employee() method to remove evaluations, and the generate_report() method to generate comprehensive performance reports.

**Features and Functionality:**

The program offers several key features to facilitate employee performance evaluation.

Adding Employee Evaluations: Users can input the employee ID and performance rating to add new evaluations. This feature allows for the continuous recording of employee performance data.

Searching for Employee Evaluations: The program enables users to search for specific employee evaluations based on the employee ID or other search criteria. This functionality provides quick access to individual performance records.

Removing Employee Evaluations: Users have the option to remove employee evaluations from the system. This feature allows for the management and updating of performance data as needed.

Generating Performance Reports: The program can generate comprehensive performance reports that summarize employee performance. These reports can include details such as employee ID, average ratings, and any other relevant information. This feature aids in decision-making and performance analysis.

Each feature plays a vital role in facilitating employee performance evaluation by providing efficient data management, easy access to evaluation records, and generating insightful reports for analysis and decision-making.

**Program:**

```
import matplotlib.pyplot as plt


class EmployeePerformanceEvaluator:
    def __init__(self, filename):
        self.filename = filename


    def add_employee(self):
        employee_id = input("Enter employee ID: ")
        employee_name = input("Enter employee name: ")
        employee_designation = input("Enter employee designation: ")
        performance_rating = input("Enter performance rating: ")


        with open(self.filename, 'a') as file:
            file.write(f"{employee_id},{employee_name},{employee_designation},{p
erformance_rating}\n")


        print("Employee evaluation added successfully.")


    def search_employee(self, employee_id):
        with open(self.filename, 'r') as file:
```

```python
            for line in file:
                data = line.strip().split(',')
                if data[0] == employee_id:
                    return data[1], data[2], data[3]

        return None

    def remove_employee(self, employee_id):
        with open(self.filename, 'r') as file:
            lines = file.readlines()

        with open(self.filename, 'w') as file:
            for line in lines:
                data = line.strip().split(',')
                if data[0] != employee_id:
                    file.write(line)

        print(f"Employee with ID {employee_id} removed successfully.")

    def generate_report(self):
        employee_ratings = {}

        with open(self.filename, 'r') as file:
            for line in file:
                data = line.strip().split(',')
                employee_id = data[0]
            employee_name = data[1]

            employee_designation = data[2]

            rating = int(data[3])  # Convert rating to integer


            if employee_id in employee_ratings:

                employee_ratings[employee_id].append(rating)

            else:

                employee_ratings[employee_id] = [rating]


        print("Employee Performance Report:")

        for employee_id, ratings in employee_ratings.items():

            try:

                average_rating = sum(ratings) / len(ratings)

                print(f"Employee ID: {employee_id}, Name: {data[1][0]}, Designation:
{data[2][0]}, Average Rating: {average_rating:.2f}")

            except ValueError:

                print(f"Invalid rating for employee {employee_id}. Skipping
calculation.")

        print(employee_ratings)


        # Plotting the performance graph

        x = list(employee_ratings.keys())

        y = [sum(ratings) / len(ratings) for ratings in employee_ratings.values()]


        plt.bar(x, y)

        plt.xlabel('Employee ID')

        plt.ylabel('Average Rating')
        plt.title('Employee Performance Report')
        plt.show()



# Example usage
evaluator = EmployeePerformanceEvaluator('employee1.txt')

while True:
    print("1. Add Employee Evaluation")
    print("2. Search Employee Evaluation")
    print("3. Remove Employee Evaluation")
    print("4. Generate Performance Report")
    print("5. Exit")
    choice = input("Enter your choice: ")

    if choice == '1':
        evaluator.add_employee()
        print('---------------------------------------\n')
    elif choice == '2':
        employee_id = input("Enter employee ID to search: ")
        result = evaluator.search_employee(employee_id)
        if result is not None:
            employee_name, employee_designation, rating = result
            print(f"Employee ID: {employee_id}, Name: {employee_name},
Designation: {employee_designation}, Performance Rating: {rating}")
            print('---------------------------------------\n')
        else:
            print("No data found for the employee.")
            print('--------------------------------------\n')
    elif choice == '3':
        employee_id = input("Enter employee ID to remove: ")
        result = evaluator.search_employee(employee_id)
        if result is not None:
            employee_name, employee_designation, _ = result
            evaluator.remove_employee(employee_id)
            print(f"Employee with ID {employee_id}, Name: {employee_name},
Designation: {employee_designation} removed successfully.")
            print('--------------------------------------\n')
        else:
            print("No record of the employee")
            print('--------------------------------------\n')

    elif choice == '4':
        print('--------------------------------------\n')
        evaluator.generate_report()
        print('--------------------------------------\n')
    elif choice == '5':
        break
    else:
        print("Invalid choice. Please try again.")
        print('--------------------------------------\n')
```