Employee Data Management System

Project Description:

The Employee Data Management System is a Python-based application designed to manage employee data for a company or organization efficiently. The system allows users to perform essential operations like:

1. Adding a new employee.
2. Viewing all employees.
3. Updating an employee's details.
4. Deleting an employee.
5. Searching for an employee by their unique ID.
6. Saving and retrieving employee data using a CSV file.

This project showcases the use of core Python concepts, Object-Oriented Programming (OOP) principles, and file handling to build a functional and scalable application.

Features of the System:

1. Add Employee:
    - Enter employee details like `ID`, `Name`, `Position`, `Salary`, and `Email`.
    - Save the details to a CSV file for persistent storage.
2. Update Employee:
    - Modify existing employee details.
    - Ensure only the required fields are updated without affecting others.
3. Delete Employee:
    - Remove an employee's data using their unique ID.
    - Automatically update the CSV file after deletion.
4. Search Employee:
    - Retrieve details of a specific employee using their ID.
5. List All Employees:
    - Display all employees with their details in a clear format.
6. File Handling:
    - Store employee data in a CSV file for long-term access.
    - Load data from the file when the program starts.

Technical Requirements:

1. Python Basics:
    - Variables, loops, conditionals, and functions.
2. Object-Oriented Programming (OOP):
    - Classes and objects.
    - Encapsulation for managing employee data.
3. File Handling:
    - Use the `csv` module for reading and writing data to CSV files.
4. Error Handling:
    - Basic validation for input data like checking valid emails or numeric salary values.

Project Structure:

1. Class `Employee`:
   Represents a single employee with attributes (`ID`, `Name`, `Position`, etc.) and methods to update or display their information.
2. Class `EmployeeManager`:
    - Handles the CRUD (Create, Read, Update, Delete) operations.
    - Interacts with the CSV file to persist data.
3. Command-Line Interface (CLI):
   A simple menu-driven interface allowing users to interact with the system.

How It Works:

1. Start the Program:
   The user is presented with a menu of actions (add, update, delete, search, list, exit).
2. Perform an Action:
   Depending on the selected option, the program performs the corresponding task (e.g., adding or updating an employee).
3. Save Data:
   Changes are saved to a CSV file, ensuring the data is persistent even after the program is closed.
4. Retrieve Data:
   Employee details are loaded from the CSV file each time the program starts.

Grading Criteria for the Project

1. **Functionality (50 points)**
    - Menu Options (10 points):
      Verify that the main menu displays all options (Add, Update, Delete, Search, List, Exit) and correctly accepts user input.
    - Add Employee (10 points):
      Check if the program successfully adds a new employee and saves the details in the CSV file.
    - Update Employee (10 points):
      Confirm the program allows users to update specific fields of an employee and reflects the changes correctly.
    - Delete Employee (10 points):
      Ensure employees can be deleted by their ID, and the CSV file updates correctly.
    - Search Employee (10 points):
      Validate the search functionality retrieves the correct employee or returns "not found" if the ID doesn’t exist.

---

2. **Code Quality (20 points)**
    - Readability (5 points):
      Check for clear variable names, organized code structure, and proper use of comments.
    - Efficiency (5 points):
      Evaluate if the program avoids unnecessary computations (e.g., iterating only when required).
    - Modularity (5 points):
      Ensure the code uses functions and methods effectively without redundant logic.
    - Error Handling (5 points):
      Verify the program handles invalid input gracefully (e.g., invalid ID or non-numeric salary).

3. **Use of OOP Principles (20 points)**
    - Class Design (10 points):
      Check if the `Employee` and `EmployeeManager` classes are designed properly, encapsulating relevant data and logic.
    - Reusability (5 points):
      Assess if the code can be easily extended (e.g., adding more features without refactoring the entire codebase).
    - Encapsulation & Abstraction (5 points):
      Confirm if the program uses proper encapsulation (e.g., methods for accessing/updating employee data) and hides unnecessary implementation details.

---

4. **File Handling (10 points)**
    - CSV Integration (5 points):
      Ensure the program correctly reads and writes employee data to a CSV file.
    - Data Persistence (5 points):
      Validate that changes (add, update, delete) are retained across program runs by saving and reloading the file.

Bonus Points (Optional)

- Validation (5 points):
  If the program validates fields like `email` or ensures `salary` is numeric.
- User Experience (5 points):
  For adding a clear and user-friendly interface or instructions.

Remark: if use chatGPT you get Zero

Sample Grading Table

Criteria	Maximum Points	Earned Points	Comments
Functionality	50		
Code Quality	20		
Use of OOP Principles	20		
File Handling	10		
Bonus	10		
Total	100		