Employee Data Management System

📌Project Overview

The **Employee Data Management System** is a Python command-line application that allows users to manage employee records efficiently. The system uses **Object-Oriented Programming (OOP)** principles, stores data in memory using dictionaries, and ensures **data persistence** by reading from and writing to a CSV file.

This project is designed to reinforce core Python concepts such as: - Dictionaries and lists - File handling using the `csv` module - Functions and conditional logic - Basic Command-Line Interface (CLI) - OOP (Encapsulation, Abstraction, Reusability)

🧱Features

The system provides a menu-based interface with the following options:

1. **Add Employee**
2. Add a new employee with ID, Name, Position, Salary, and Email
3. Data is stored in memory and saved to a CSV file
4. **View All Employees**
5. Display all employees in a readable format
6. Data is loaded from memory
7. **Update Employee**
8. Update employee details using their unique ID
9. Empty fields are ignored and not updated
10. Changes are saved back to the CSV file
11. **Delete Employee**
12. Delete an employee using their ID
13. CSV file is updated accordingly
14. **Search Employee**
15. Search and display employee details by ID

16. **Exit**

17. Exit the program safely

🛠️Technologies Used

- **Python 3**
- **csv module** for file handling
- **Git & GitHub** for version control

📂Project Structure

```
Employee-Data-Management-System/
│
├── employee_manager.py   # Main Python program
├── employees.csv         # CSV file storing employee data
└── README.md             # Project documentation
```

📩How to Run the Program

1. Clone the repository:

```
git clone https://github.com/your-username/employee-data-management-system.git
```

2. Navigate to the project folder:

```
cd employee-data-management-system
```

3. Run the program:

```
python employee_manager.py
```

📖How It Works

Start the Program

- The program loads employee data from `employees.csv` into memory.
- A menu of options is displayed.

Perform an Action

- The user selects an option (Add, Update, Delete, Search, List, Exit).
- The program performs the requested operation.

Save Data

- Any changes (add, update, delete) are saved to the CSV file.

Retrieve Data

- When the program restarts, all employee data is reloaded from the CSV file.

🔗Validation & Error Handling

- Salary must be numeric
- Employee ID must be unique
- Invalid menu options are handled gracefully
- Displays appropriate messages for non-existing employee IDs

---

OOP Design

- Uses a single class: `EmployeeManager`
- Encapsulates employee data and related operations
- Easily extendable (e.g., add new fields or features)

🕐Bonus Features

- Basic input validation (salary, empty fields)
- User-friendly menu and messages

📤Submission Instructions

- Submit **only the GitHub repository link** via **Microsoft Teams**
- Do **not** upload any files directly

- Submissions outside the platform will not be accepted

💻 Author

Donia

📜 License

This project is for educational purposes only.