Assignment 2 - Employee Management System

1. Introduction

1.1 Purpose

The Employee Management System is designed to help organizations manage employees, departments, and projects efficiently. It provides a structured approach to handling employee records, assigning employees to departments, and managing projects within an organization. The system is intended to help HR departments streamline these tasks effectively.

1.2 Scope

This project will be developed using **Object-Oriented Programming (OOP) principles in Python** with a **modular programming approach** to ensure scalability and maintainability. The system will cover:

- **Employee Management:** Creating, updating, and deleting employee records.
- **Department Management:** Assigning managers and adding/removing employees.
- **Project Management:** Assigning employees to projects and updating project details.
- **HR System Management:** Overseeing employees, departments, and projects as a whole.

1.3 Project Deadline

- **Final Submission:** Thursday, 6 March 2025, at 11:59 PM (Egypt Time).

1.4 Intended Audience

- This system is part of the **Winter Training 2025** and is assigned as a mini-project for trainees to apply **OOP concepts** in a real-world scenario.

1.5 Work Schedule (2 March - 6 March 2025)

- **2 March:** Understand project requirements and set up the development environment.
- **3 March:** Design class structures and define attributes.
- **4 March:** Implement class constructors and function signatures.
- **5 March:** Develop core functionalities and integrate system components.
- **6 March:** Test and debug the system before submission.

---

2. System Overview

2.1 Modular Programming File Structure

The system follows a **modular approach**, where each functionality is placed in a separate Python file:

- **employee.py** – Handles employee details, updates, and project assignments.
- **department.py** – Manages department creation, employee management, and manager assignments.
- **project.py** – Tracks project details, employee assignments, budget, and deadlines.
- **hr_system.py** – Serves as the main controller connecting employees, departments, and projects.
- **main.py** – The entry point to test and execute the system.

2.2 Explanation of System Components

Employee Class (employee.py)

The **Employee** class represents individual employees in the organization. Each employee has specific details such as:

- **Employee ID (emp_id)** – A unique identifier for each employee.
- **Name (name)** – The full name of the employee.
- **Position (position)** – The job role or title of the employee.
- **Salary (salary)** – The employee’s salary.
- **Projects (projects)** – A list to track projects assigned to the employee.

Functions in Employee Class:

1. **Constructor (__init__)** – Initializes an employee with their ID, name, position, salary, and an empty list of projects.
2. **update_info(name, position, salary)** – Updates the employee’s information.
3. **assign_project(project_id)** – Adds a project to the employee’s list of assigned projects.
4. **remove_project(project_id)** – Removes a project from the employee’s list.
5. **get_details()** – Returns all details of the employee.

Department Class (department.py)

The **Department** class represents different departments within the organization. Each department has:

- **Department ID (dept_id)** – A unique identifier for the department.
- **Name (name)** – The name of the department (e.g., IT, HR, Marketing).
- **Manager ID (manager_id)** – The ID of the manager overseeing the department (optional).
- **Employees (employees)** – A list of employee IDs assigned to the department.

Functions in Department Class:

1. **Constructor (__init__)** – Initializes a department with its ID, name, and an optional manager. Also creates an empty list of employees.
2. **assign_manager(manager_id)** – Assigns a manager to the department.
3. **add_employee(emp_id)** – Adds an employee to the department.
4. **remove_employee(emp_id)** – Removes an employee from the department.
5. **get_department_details()** – Returns department details, including assigned employees and the manager.

Project Class (project.py)

The **Project** class represents projects that employees work on. Each project has:

- **Project ID (project_id)** – A unique identifier for the project.
- **Name (name)** – The name of the project (e.g., AI Research, Web Development).
- **Budget (budget)** – The allocated budget for the project.
- **Deadline (deadline)** – The project’s completion deadline.
- **Employees (employees)** – A list of employee IDs assigned to the project.

Functions in Project Class:

1. **Constructor (__init__)** – Initializes a project with its ID, name, budget, and deadline. Also creates an empty list for employees.
2. **assign_employee(emp_id)** – Assigns an employee to the project.

3. **remove_employee(emp_id)** – Removes an employee from the project.
4. **update_budget(budget)** – Updates the project’s budget.
5. **update_deadline(deadline)** – Changes the project deadline.
6. **get_project_details()** – Returns project details, including assigned employees.

HRSystem Class (hr_system.py)

The **HRSystem** class acts as the **central controller** that manages all employees, departments, and projects within the organization. It ensures smooth interactions between different components.

Functions in HRSystem Class:

1. **Constructor (__init__)** – Initializes empty lists to store employees, departments, and projects.
2. **add_employee(employee)** – Adds a new employee to the system.
3. **add_department(department)** – Adds a new department to the system.
4. **add_project(project)** – Adds a new project to the system.

---

3. Main Script (main.py)

```
# Import necessary classes
from employee import Employee
from department import Department
from project import Project
from hr_system import HRSystem

# Initialize the HR System
hr = HRSystem()

# Create employees
emp1 = Employee(1, "Ali", "Software Engineer", 15000)
emp2 = Employee(2, "Mona", "Data Scientist", 18000)
emp3 = Employee(3, "Kareem", "Project Manager", 25000)

# Add employees to HR system
hr.add_employee(emp1)
hr.add_employee(emp2)
hr.add_employee(emp3)

# Create departments
dept1 = Department(101, "Engineering")
dept2 = Department(102, "Data Science")

# Add departments to HR system
hr.add_department(dept1)
hr.add_department(dept2)

# Assign managers to departments
dept1.assign_manager(emp1.emp_id)
```

```python
dept2.assign_manager(emp2.emp_id)

# Add employees to departments
dept1.add_employee(emp1.emp_id)
dept1.add_employee(emp3.emp_id)
dept2.add_employee(emp2.emp_id)

# Create projects
proj1 = Project(201, "AI Chatbot", 50000, "30-Apr-2025")
proj2 = Project(202, "E-commerce Platform", 75000, "15-May-2025")

# Add projects to HR system
hr.add_project(proj1)
hr.add_project(proj2)

# Assign employees to projects
proj1.assign_employee(emp1.emp_id)
proj1.assign_employee(emp2.emp_id)
proj2.assign_employee(emp3.emp_id)

# Employees join projects
emp1.assign_project(proj1.project_id)
emp2.assign_project(proj1.project_id)
emp3.assign_project(proj2.project_id)

# Display details
print("\n--- Employees ---")
for employee in hr.employees:
    print(employee.get_details())
print("\n--- Departments ---")
for department in hr.departments:
    print(department.get_department_details())


print("\n--- Projects ---")
for project in hr.projects:
    print(project.get_project_details())
```

Expected Output:

```
--- Employees ---
{'ID': 1, 'Name': 'Ali', 'Position': 'Software Engineer', 'Salary': 15000,
'Projects': [201]}
{'ID': 2, 'Name': 'Mona', 'Position': 'Data Scientist', 'Salary': 18000,
'Projects': [201]}
{'ID': 3, 'Name': 'Kareem', 'Position': 'Project Manager', 'Salary': 25000,
'Projects': [202]}


--- Departments ---
{'ID': 101, 'Name': 'Engineering', 'Manager': 1, 'Employees': [1, 3]}
{'ID': 102, 'Name': 'Data Science', 'Manager': 2, 'Employees': [2]}


--- Projects ---
{'ID': 201, 'Name': 'AI Chatbot', 'Budget': 50000, 'Deadline': '30-Apr-2025',
'Employees': [1, 2]}
{'ID': 202, 'Name': 'E-commerce Platform', 'Budget': 75000, 'Deadline': '15-
May-2025', 'Employees': [3]}
```

Good luck!