■ Project Title: Employee Management System

■ Project Overview:

This project is a Python Object-Oriented Programming (OOP)–based system that models a real-world company structure. It manages employees, developers, and managers, showing how inheritance, encapsulation, and object relationships work in a real-world way. In short — this project shows how a manager handles multiple employees, developers have their own programming skills, and all of them share common employee attributes like name, salary, and email.

■ Objective:

The main objective is to demonstrate OOP concepts like classes, objects, inheritance, and polymorphism. It aims to create a mini system that can manage employees and managers in a structured way, showing how different employee types (like normal employees and developers) can be connected through class inheritance.

■■ Concepts Used:

Concept	Explanation
Class & Object	Classes are blueprints; objects are instances (Employee, Developer, Manager).
Inheritance	Developer and Manager inherit from the Employee class.
Encapsulation	Each object stores its own data (like name, salary, email).
Polymorphism	Methods like fullname() work for all types of employees.
Composition	Manager has a list of employee objects — a 'has-a' relationship.

■ Working of the Program:

The system is divided into three main classes: Employee, Developer, and Manager. 1. Employee Class – Base class with details like first name, last name, salary, and email. It has methods to show full name and increment salary. 2. Developer Class – Inherits from Employee and adds an extra attribute 'prog_lang' (programming language). 3. Manager Class – Inherits from Employee and maintains a list of employees it manages. It can add, remove, or display employees. Execution Flow: Objects for employees, developers, and a manager are created. The manager object manages multiple employees (and later developers), and the program displays who is managed by the manager.

■ Example Output (Short Version):

Manager: Swaranjali Shingate manages the following employees: ---> Saniya Shaikh ---> Snehal Shinde ---> Swara Shingate After adding developers: ---> Tanu Mahale ---> Yashshri Patil

■ Project Outcome:

This project demonstrates real-world modeling using OOP. It helps understand class relationships (inheritance and composition) and builds a strong foundation for larger systems like HR, Payroll, or CRM software.

■ Conclusion:

The Employee Management System provides a clear understanding of OOP implementation in Python. It models how a real company structure operates using inheritance and encapsulation. The project is simple, readable, and scalable — forming a strong base for more advanced applications.