SOLID Principle

Real-Time Example: Employee Management System (SOLID Principles in C#)

Imagine an **Employee Management System** in a company where we handle **employee details, salary calculations, and report generation**.
Initially, the system has **tightly coupled code**, making it **hard to maintain and extend**.
We will **refactor** it using the **SOLID principles** for **better design and maintainability**.

An **Employee Management System (EMS)** is a software solution that helps organizations manage their workforce efficiently. It handles **employee records, salary calculations, and report generation** while ensuring flexibility and scalability.

In this article, we will design an **Employee Management System in C#**, following the **SOLID principles** to make the system **modular, maintainable, and extensible**.

Problem Statement

- A company requires an **Employee Management System** with the following functionalities:
- Store and manage **employee details** (name, ID, etc.).
- Calculate **salaries** based on employee type (Full-time, Intern, Contract-based, etc.)
- Generate **reports** in different formats (PDF, Excel).
- Allow easy **modifications and extensions** to support new requirements without affecting existing functionality.

- However, a **monolithic and tightly coupled design** can lead to:
- **Difficulty in adding new salary structures.**
- **Code duplication across multiple components.**
- **Hard-to-maintain report generation logic.**
- **Challenges in unit testing due to high dependencies.**

To address these challenges, we will refactor the system using **SOLID design principles**, ensuring **better structure, flexibility, and testability**.

Key Features of the Employee Management System

1. Employee Data Management

- Store essential details such as **Name, ID, and Employment Type**.
- Extend the system to support different categories of employees (e.g., Full-time, Interns, Contractors).

2. **Salary Calculation**

- Implement **different salary structures** based on employee type.
- Allow **new salary types** (e.g., hourly-based, commission-based) **without modifying existing code**.

3. Report Generation

- Generate **reports in multiple formats** (PDF, Excel).
- Ensure **scalability** by enabling easy integration of new reporting methods.

4. Extensibility and Maintainability

- Use **interfaces and abstraction** to make the system flexible.
- Ensure **loosely coupled components**, so new features can be added without breaking the existing system.