**SOLID Principles**

**What are SOLID Principles:**

They are five design guidelines intended to make software design more understandable, flexible, and maintainable. These principles were introduced by Robert C. Martin and are foundational in Object-Oriented design (OOD), they help in avoiding software design issues that can lead to rigid, fragile, and hard-to-maintain code.

**The SOLID Stands for:**

1. **S** – Single Responsibility Principle (SRP).
2. **O** – Open/Closed Principle (OCP).
3. **L** – Liskov Substitution Principle (LSP).
4. **I** – Interface Segregation Principle (ISP).
5. **D** – Dependency Inversion Principle (DIP).

**Single Responsibility Principle**

A class should have only one reason to change, meaning it should have only one job or responsibility.

This principle helps to ensure that each class or module in you system does one thing and does it well. If class has more than one responsibility, changes to one responsibility may affect the others, making the class more difficult to maintain.

**Open Closed Principle (OCP)**

Software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification.

This means that you should be able to extend the behavior of a class without modifying its source code. It’s a way of ensuring that changes don’t break existing code, allowing for greater flexibility and less risk when introducing new functionality.

In software OCP encourages you to extent the functionality of your class without altering their original structure, much like extending your house without knocking it down.

And ensures that the core class (the app) doesn’t need to be modified for every new feature (plugin).