**Single Responsibility Principle Explanation with Example C# code:**

You probably have heard about SOLID principles: single responsibility, open-closed, liskov substitution, interface segregation and dependency inversion. We will discuss single responsibility in this article.

The Single Responsibility Principle (SRP) is the concept that any single object in object-oriented programming (OOP) should be made for one specific function. When a class has more than one responsibility, there are also more reasons to change that class. This kind of code is not appropriate for single responsibility principle.

Classes, software components and microservices that have only one responsibility are much easier to explain, understand and implement than the ones that provide a solution for everything. This reduces the number of bugs and improves development speed.

“A class should have one, and only one, reason to change.”

-Robert C. Martin

**We will write a code without considering single responsibility principle and see how important this principle is:**

interface userinterface

{

bool login(string username, string password);

bool register(string username, string password, string email);

void loginerror(string error);

bool sendemail(string emailcontent);

}

This is a bad example of single responsibility principle because UserInterface does all the job itself. We should use different interfaces for different jobs.

interface UserInterface

{

//Login and Register:

bool Login(string username, string password);

bool Register(string username, string password, string email);

}

interface LoginInterface

{

//Error:

void LoginError(string error);

}

interface EmailInterface

{

//Send Email:

bool SendEmail(string emailContent);

}